ISSN 1064-5624, Doklady Mathematics, 2023, Vol. 108, Suppl. 2, pp. S215–S225. © Pleiades Publishing, Ltd., 2023.
ISSN 1064-5624, Doklady Mathematics, 2023. © Pleiades Publishing, Ltd., 2023.
Russian Text © The Author(s), 2023, published in Doklady Rossiiskoi Akademii Nauk. Matematika, Informatika, Protsessy Upravleniya, 2023, Vol. 514, No. 2, pp. 60–71.

Deep Metric Learning: Loss Functions Comparison

R. L. Vasilev[a],* and A. G. D'yakonov[a],**

Received June 30, 2023; revised September 19, 2023; accepted October 15, 2023

Abstract—An overview of deep metric learning methods is presented. Although they have appeared in recent years, these methods were compared only with their predecessors, with neural networks of outdated architectures used for representation learning (representations on which the metric is calculated). The described methods were compared on different datasets from several domains, using pre-trained neural networks comparable in performance to SotA (state of the art): ConvNeXt for images and DistilBERT for texts. Labeled datasets were used, divided into two parts (train and test) so that the classes did not overlap (i.e., for each class its objects are fully in train or fully in test). Such a large-scale honest comparison was made for the first time and led to unexpected conclusions, viz. some "old" methods, for example, Tuplet Margin Loss, are superior in performance to their modern modifications and methods proposed in very recent works.

Keywords: machine learning, deep learning, metric, similarity

DOI: 10.1134/S1064562423701053

1. INTRODUCTION

Many data analysis and machine learning problems require estimating the proximity of objects such as for the search over images [1] or text documents [2], face recognition [3], and identification [4]. Modern methods for solving such problems use deep metric learning. Metric learning is assumed to mean obtaining an automatic way to calculate distances (for example, in the form of the output of a neural network), in which the distances between objects that should be similar (for example, they belong to the same class) are small, and the distances between objects that must differ are great.

We describe this idea more formally. Consider a set of objects X and a set of labels Y. Objects $x \in X$ can be of any kind (images, texts, audio, tabular data, etc.), and each object can have its corresponding label $y \in Y$. If objects are indexed, we assume that the object x_t is matched to the label y_t. The general metric learning problem is to build a metric $D_\theta(\cdot, \cdot) : X \times X \to \mathbb{R}$ out of the parametric family $\{D_\theta\}$ so that it satisfies the property

$$D_\theta(x_1, x_2) \ll D_\theta(x_1, x_3)$$

for $y_1 = y_2 \neq y_3$. Not all problems have object labels; however, there is often a way to construct a pair (x_1, x_2) of objects that can be logically considered similar such as an object and its augmentation:

• a signal and its noisy copy;

• an image and its rotated copy;

• a text and the same text with some words replaced by synonyms.

Such pairs are called *positive* while the others are called *negative*, for example a random pair of sample objects. Formally, for the set of positive pairs P and negative pairs N, we can write a system of inequalities

$$D_\theta(x_i, x_j) < D_\theta(x_t, x_s), \quad (i, j) \in P, \quad (t, s) \in N,$$

but there are a large number of inequalities in such a system, in addition, the sign $\ll$ in the property we described means the desire for these distances to be significantly different, so one uses the approaches that we describe below.

The deep metric learning problem is to train a neural network, i.e., the mapping $f_\theta(\cdot) : X \to \mathbb{R}^n$, wherein

$$D_\theta(x_1, x_2) = D(f_\theta(x_1), f_\theta(x_2)),$$

where D is the fixed simple metric in the finite-dimensional space $\mathbb{R}^n$ of vector representations (embeddings) of objects, the Euclidean distance being used more often. It is also common to limit the parameterization by the unit sphere $\|f_\theta(x)\| = 1$ and introduce the scalar product

$$S_\theta(x_1, x_2) = \langle f_\theta(x_1), f_\theta(x_2) \rangle,$$

rather than $D_\theta(x_1, x_2)$; here, one strives to make

$$S_\theta(x_1, x_2) \gg S_\theta(x_1, x_3)$$

[a]Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Moscow, Russia
*E-mail: artnitolog@yandex.com
**E-mail: djakonov@mail.ru

hold for $y_1 = y_2 \neq y_3$.

To find the optimal parameters θ, the loss function L is given and optimized on the training set. Further, we consider particular loss functions.

Metric learning is useful in supervised learning problems where there are quite a lot of classes and they are poorly represented. Also, it is when not all classes are in the training set and the algorithm must detect the appearance of objects from unknown classes (open-world classification).

In this work, we try to systematize neural network approaches to metric learning and perform a great number of experiments to compare different loss functions in different domains, viz. images and texts. This is the first time such a large-scale comparison has been made. Section 2 discusses the principal evaluation metrics, allowing us to compare methods with each other. Section 3 provides an overview of various methods such as the ones based on comparing representations or on reducing it to the classification problem. Section 4 describes experiments with these methods on various datasets, neural network architectures and modalities.

2. EVALUATION METRICS

2.1. Search Related

An object, for which the nearest neighbors are searched for, is called a *query*, and the objects, over which the search is performed, are called *documents*; the terminology comes from information retrieval. Upon the query, the information retrieval algorithm yields an ordered list of documents; this is when one can use a trained metric and arrange documents in the order of their increasing distance from the query. Moreover, among all documents there is a subset of *relevant* documents, viz. those that we would like to retreive, and they should be at the top of our list. The following evaluation metrics assess how well this works.

Recall@K is one of the most commonly used evaluation metrics in metric learning and is defined as the proportion of queries for which a relevant document was found among K nearest neighbors. Note that this definition of Recall@K is used specifically in metric learning problems; in information retrieval, Recall@K is usually understood as the proportion of the found objects among the relevant ones.

MAP (mean average precision) is another frequently used evaluation metric. We define Precision@K as the proportion of the relevant objects among K found ones. Let the output contain indexes of the relevant objects $K_1, \ldots, K_R$, then we can calculate

$$\mathrm{AP} = \frac{1}{R}\sum_{i=1}^{R} \mathrm{Precision@}K_i$$

and MAP is AP averaged over all queries.

R-Precision is also frequently calculated; it is equal to Precision@R, where R is the total number of relevant documents. In [5], a functional is proposed that combines the ideas of R-Precision and MAP. For a separate query with R relevant documents, it equals

$$\mathrm{AP@R} = \frac{1}{R}\sum_{i=1}^{R} \mathrm{Precision@i},$$

the overall MAP@R is obtained for the entire dataset by averaging over all queries.

Also, one measures the *mean reciprical rank* (MRR)

$$MRR = \frac{1}{m}\sum_{i=1}^{n} \frac{1}{\mathrm{rank}_i},$$

where rank_i is the position number of the first relevant document, and m is total number of queries.

2.2. Cluster Related

In addition to search quality criteria, metric learning often evaluates the quality of clustering in the space $\mathbb{R}^n$ of representations $\{f_\theta(x) | x \in X\}$ as representations of similar objects are expected to form clusters. The disadvantage of "cluster" evaluation metrics based on this idea is that the functionals depend on the selected clustering algorithm.

Let $U = \{U_1, \ldots, U_n\}$ be the true partition X into groups of similar objects, and $V = \{V_1, \ldots, V_n\}$ be the one obtained by the clustering algorithm (in this work, the experiments use k-means [6]) on $f_\theta(X)$. Then *mutual information* is

$$\begin{aligned} MI(U,V) &= \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P_{UV}(i,j) \log \frac{P_{UV}(i,j)}{P_U(i)P_V(j)} \\ &= KL(P_{UV} \,\|\, P_U P_V), \end{aligned}$$

where $P_U(i) = \frac{|U_i|}{|U|}$ is the estimate of the probability of a random object to fall into the ith partition cluster U; similarly, $P_V(i) = \frac{|V_i|}{|V|}$ and $P_{UV}(i,j) = \frac{|U_i \cap V_j|}{|V|}$ is the estimate of the probability to fall into U_i and V_j simultaneously.

The entropy of partition S is defined as follows

$$H(S) = -\sum_{k=1}^{|S|} P_S(k) \log P_S(k).$$

There exist several normalizations: NMI (normalized mutual information) and AMI (adjusted mutual information). NMI is calculated as follows

$$NMI(U,V) = \frac{MI(U,V)}{\frac{1}{2}(H(U) + H(V))},$$

however, AMI

$$AMI(U,V) = \frac{MI(U,V) - \mathbb{E}MI(U,V)}{\frac{1}{2}(H(U) + H(V)) - \mathbb{E}MI(U,V)},$$

is used more often, where $\mathbb{E}MI(U,V)$ is the mathematical expectation of $MI(U,V)$ over all possible partitions U and V.

3. METHODS OF DEEP METRIC LEARNING

3.1. Methods Based on Representation Comparison

We describe the methods that are used to train the metric and are based on comparing object representations in the value space $\{f_\theta(x) | x \in X\}$. Each method is defined by a loss function used to optimize the neural network that defines the mapping $f_\theta(x)$.

3.1.1. Contrastive Loss. One of the first loss functions proposed for metric learning was *Contrastive Loss* [7], which is defined as follows

$$L = \mathbb{I}\{y_i = y_j\}[D_\theta(x_i, x_j) - m_p]_+^2 + \mathbb{I}\{y_i \neq y_j\}[m_n - D_\theta(x_i, x_j)]_+^2, \quad (1)$$

here and in what follows $\mathbb{I}\{A\} = 1$ if and only if the expression A is true, $[z]_+ = \max\{z, 0\}$ is the cut-off function. Functional (1) penalizes positive pairs if the distance between them is greater than a certain threshold m_p (most often assumed to equal zero) while it penalizes the negative pairs until the distance exceeds the threshold m_n.

3.1.2. Triplet Loss. In [8], it is proposed to compare triplets of objects (x_a, x_p, x_n), where $y_a = y_p$ yet $y_a \neq y_n$, rather than their pairs

$$L = [D_\theta(x_a, x_p)^2 - D_\theta(x_a, x_n)^2 + m]_+. \quad (2)$$

The main difference between (1) and (2) is that Contrastive Loss penalizes the absolute distance within a single pair of objects while Triplet Loss limits the *difference* between *positive* and *negative* pairs in triplets. Triplet Loss is often used in the face recognition problem [8], with its modifications used in other domains, for example, for training text representations [2].

3.1.3. Fast AP. As we note in Section 2, MAP is one of the principal evaluation metrics in metric learning problems. In [9], it is proposed to optimize the AP approximation. The principal issue for the functional from an optimization point of view is the sorting operation, for which gradient methods cannot be applied. The authors' idea is to interpret AP as the area under the PR curve and consider Precision and Recall as parametric functions of the distance between the queries and objects. The authors propose the following approximation

$$\text{FastAP} = \frac{1}{N_q^+} \sum_{j=1}^{L} \frac{H_j^+ h_j^+}{H_j}. \quad (3)$$

• Representations are assumed to be l_2-normalized, with $\|f_\theta(x)\| = 1$—in this case, the possible distances between the query and the retrieval candidates belong to the segment [0, 2].

• The segment [0, 2] is divided into bins $\{z_1, \ldots, z_L\}$ (number of bins L is a hyperparameter), and h_j is the number of objects included in the jth bin. $H_j = \sum_{k=1}^{j} h_k$ is the cumulative sum.

• h_j^+ and H_j^+ are the same counters, but only for objects relevant to the query; N_q^+ is the total number of relevant objects.

• In fact, in (3) *histogram binning* is used, which in this case approximates the true distribution (viz., the distribution density and function) of distances by piecewise constant functions

$$h(z) = \sum_{j=1}^{L} h_j \cdot \mathbb{I}\{z \in z_j\}, \quad H(z) = \sum_{j=1}^{L} H_j \cdot \mathbb{I}\{z \in z_j\}.$$

• Since the *piecewise constant functions* are non-differentiable, we replace $h(z)$ and $H(z)$ by linear interpolation during optimization, which leads to continuous *piecewise linear functions*, with their gradients defined and constant within individual bins. Such relaxation was proposed in [10].

3.1.4. Centroid Triplet Loss. In [11], the authors propose to modify (2) by replacing positive and negative objects (x_p, x_n) with respect to x_a by the centers of their classes (c_a, c_p)

$$L = [D_\theta(x_a, c_p)^2 - D_\theta(x_a, c_n)^2 + m]_+.$$

During training, class centers are counted by batch (and the batches themselves should be made large), and x_a is excluded. The principal motivation of this method is to simplify the application stage, viz. the search can be carried out only for the centers pre-calculated in advance (this time for the entire sample) rather than for all objects.

3.1.5. Margin Loss. Authors of [12] suggest using

$$L = [\alpha + (2 \cdot \mathbb{I}\{y_i = y_j\} - 1) \cdot (D_\theta(x_i, x_j) - \beta)]_+. \quad (4)$$

In fact, (4) differs from (1) by the square of the Euclidean metric replaced by the metric itself $(l_2^2 \mapsto l_2)$ and by the modified parameterization $m_p = \beta - \alpha$, $m_n = \beta + \alpha$. Thus, β corresponds to the boundary between *positive* and *negative* pairs, and α corresponds to the required distance from this boundary.

3.1.6. Multi Similarity Loss. In [13], it is proposed to use more information about the distances between objects in the batch in the loss function

$$L = \sum_{i=1}^{m}\left\{\frac{1}{\alpha}\log\left[1+\sum_{k\in P_i} e^{-\alpha(S_{ik}-\lambda)}\right] + \frac{1}{\beta}\log\left[1+\sum_{k\in N_i} e^{\beta(S_{ik}-\lambda)}\right]\right\}. \tag{5}$$

Here, $S_\theta(x_i, x_j) = \langle f_\theta(x_i), f_\theta(x_j)\rangle$ is parameterized. Note that expression (5) can be considered as a smooth approximation of (1) (with additional coefficients) that includes all pairwise distances within the batch.

3.1.7. SNN Loss. The loss function based on distances within a batch can also be defined as follows [14]

$$L = -\frac{1}{m}\sum_{i=1}^{m}\log\frac{\sum_{j=1}^{m}\mathbb{I}\{y_i = y_j\}\exp\{S_\theta(x_i,x_j)/\tau\}}{\sum_{j=1}^{m}\exp\{S_\theta(x_i,x_j)/\tau\}}, \tag{6}$$

τ is the temperature (it can be either a customized parameter or a hyperparameter).

3.1.8. SupCon Loss. In [15], the authors noticed that there are different ways to aggregate positive examples in (6) and proposed the following alternative

$$L = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{m}\mathbb{I}\{y_i = y_j\}\log\frac{\exp\{S_\theta(x_i,x_j)/\tau\}}{\sum_{j=1}^{m}\exp\{S_\theta(x_i,x_j)/\tau\}}. \tag{7}$$

Option (7) is usually better optimized than (6).

3.1.9. SNR Loss. In [16], (1) is used, but instead of the Euclidean metric the authors optimize SNR (signal-to-noise ratio)

$$D_\theta(x_i, x_j) = \frac{1}{\mathrm{SNR}} = \frac{\mathrm{Var}[f_\theta(x_i) - f_\theta(x_j)]}{\mathrm{Var}[f_\theta(x_i)]},$$

here Var is the sample variance. Unlike the Euclidean metric, the SNR distance is not symmetric, so the order of the elements in the pair matters, similar to how Triplet Loss (2) is affected by the replacement $x_a \leftrightarrow x_p$.

3.1.10. Tuplet Margin Loss. Authors of [17] made several significant changes to (2)

$$L = \log\left(1+\sum_{i=1}^{k-1} e^{s(\cos\theta_{a,n_i} - \cos(\theta_{a,p}-\beta))}\right). \tag{8}$$

• In (8), the number of negative objects is not limited by the triplet, viz. for each positive pair (x_a, x_p) from the remaining classes, one negative example is sampled, forming the set

$$(x_a, x_p, x_{n_1}, \ldots, x_{n_{k-1}}).$$

• Representations on the sphere are used ($\|f_\theta(x_i)\| = 1$), therefore the cosine of the angle between the vectors is calculated as a scalar product.

• The coefficient $\beta \geqslant 0$ is used to combat overtraining on hard triplets (triplets in which $\theta_{a,n_i} \ll \theta_{a,p}$), which, with $\beta = 0$, make a much larger contribution to the loss function as compared to other elements of the set.

• Instead of the cut-off function $[z]_+ = \max\{z, 0\}$, its differentiable approximation Softplus(z) = log(1 + e^{sz}) is used, where s is the scale factor (responsible for the radius of the hypersphere and affecting the speed of convergence).

3.1.11. Circle Loss. In [18], it is proposed to optimize the modified relaxation (1)

$$L = \log\left[1+\sum_{j=1}^{L} e^{\gamma[s_n^j - m_n]_+ s_n^j}\sum_{i=1}^{K} e^{-\gamma[m_p - s_p^i]_+ s_p^i}\right], \tag{9}$$

where s_p^j and s_n^i are similarity functionals between positive and negative pairs.

3.2. Classification Based Methods

This group of methods differs by the fact that a sample labeled into a fixed number of classes c is used for training. In fact, the classification problem is solved, but classification methods do not ensure obtaining the space of representations with objects of each class being well grouped.

3.2.1. ArcFace. In ArcFace [19], a loss function is proposed, by optimizing which the authors managed to obtain high-quality vector representations in the face recognition problem. The idea is to consider Softmax-loss on the unit sphere and use the geodesic distance.

We introduce an auxiliary weight matrix $W \in \mathbb{R}^{n\times c}$. When solving a classification problem, one could optimize Softmax-loss (cross-entropy)

$$-\log\frac{e^{W_{y_i}^T f_\theta(x_i)}}{\sum_{j=1}^{c} e^{W_j^T f_\theta(x_i)}}. \tag{10}$$

We consider the jth logit (10): $W_j^T f_\theta(x_i) = \|W_j\|\,\|f_\theta(x_i)\|\cos_{\theta_j}$, where θ_j is the angle between W_j and $f_\theta(x_i)$. We fix the norm of all weights $\|W_j\| = 1$ and normalize the space of representations $\|f_\theta(x_i)\| = s$; then, (10) takes the form (11)

$$L = -\log\frac{e^{s(\cos(\theta_{y_i}))}}{e^{s(\cos(\theta_{y_i}))} + \sum_{j\neq y_i} e^{s\cos(\theta_j)}}, \tag{11}$$

if we add the margin to the angle in (11), we get ArcFace Loss [19]

$$L = -\log \frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))} + \sum_{j \neq y_i} e^{s\cos(\theta_j)}}. \quad (12)$$

The parameter m in this case is responsible for both intraclass compactness and interclass separability.

3.2.2. CosFace. Almost the same idea was proposed in [20], only with the margin used for cosines rather than for angles

$$L = -\log \frac{e^{s(\cos(\theta_{y_i})+m)}}{e^{s(\cos(\theta_{y_i})+m)} + \sum_{j \neq y_i} e^{s\cos(\theta_j)}}. \quad (13)$$

Although methods (12) and (13) often have almost the same quality [19], ArcFace often performs slightly better in the face recognition problem.

3.2.3. SubCenter ArcFace. In [21], it is proposed to strengthen the robustness of functional (12) to noise. To do this, it is proposed to use K subcenters instead of one for each class. The form of loss function (12) is the same, but K weight matrices $W^1, \ldots, W^K \in \mathbb{R}^{n \times c}$ are introduced

$$\theta_j = \arccos\left(\max_{k \in 1,\ldots,K} \{(W^k)_j^T f_\theta(x_i)\}\right). \quad (14)$$

Thus, the closest one is selected among K subcenters in (14). With this approach, as shown in [21], most of the objects of a particular class will be accumulated in the neighbourhood of one subcenter, while noise objects will be assigned with other non-dominant (containing a small number of representatives) subcenters that can be discarded at the inference stage.

3.2.4. SoftTriple Loss. In [22], the idea of several centers per class is also being considered.

$$L = -\log \frac{e^{s(G(i,y_i)+m)}}{e^{s(G(i,y_i)+m)} + \sum_{j \neq y_i} e^{sG(i,j)}},$$

where G is the function of proximity between the ith object and the class c, and w_c^k is the kth center of the class c

$$G(i,c) = \sum_k \frac{\exp\left\{\frac{1}{\gamma}\langle f_\theta(x_i), w_c^k\rangle\right\}}{\sum_{k'} \exp\left\{\frac{1}{\gamma}\langle f_\theta(x_i), w_c^{k'}\rangle\right\}} \langle f_\theta(x_i), w_c^k\rangle.$$

3.2.5. Proxy-Anchor Loss. The authors of [23] propose adding special proxy objects and optimize the distance between them and objects of the same class. Some of the methods above are like this—for example, in (12) the class centers θ_j can be considered as proxy objects. In [23], the following loss function is used

$$L = \frac{1}{|P^+|} \sum_{p \in P^+} \log\left(1 + \sum_{x \in X_p^+} e^{-\alpha(S(x,p)-m)}\right) + \frac{1}{|P|} \sum_{p \in P} \log\left(1 + \sum_{x \in X_p^-} e^{-\alpha(S(x,p)-m)}\right),$$

where α is the scale hyperparameter, m is the shift hyperparameter, P is the set of all proxy objects, P^+ is the set of positive ones (in the batch), X_p^-, X_p^+ and X_p^- are the partition of the batch with respect to proxy objects.

4. EXPERIMENTS

As we saw in the previous section, there are quite a lot of loss functions for solving the metric learning problem. Often, when choosing a specific method, one checks whether a ready-made implementation is available or how popular the method is. In this work, we decided to implement and test all the described methods, ensuring fair assessment and comparison of their quality.

4.1. Data

One of the most important features of metric learning is that metric properties (similar objects are close, objects of different classes are not) are preserved on new data. To fairly measure quality in Supervised data, training and testing are performed on non-overlapping classes.

This work uses six datasets, viz. four with images and two text ones, the objects in them are divided into training and test sets according to Table 1.

4.1.1. Cars-196. The initial Cars-196 dataset [24] contains 16 185 photos divided into 196 classes. Each photo shows a car, and the class is characterized by its brand, year and model (color, angle, background, etc. may vary). Examples of images are shown in Fig. 1. This is one of the most frequently used datasets in measuring the quality of Metric Learning methods.

4.1.2. CUB-200. The Caltech-UCSD Birds 200 dataset [25] contains 11 788 photos of birds, a total of 200 categories, examples are given in Fig. 2.

4.1.3. SOP. The Stanford Online Products dataset [26] contains 120 053 images of various products, the total number of categories is 22 634, examples are given in Fig. 3.

4.1.4. Dogs-130. The Tsinghua Dogs dataset [27] contains a total of 70 432 different photos of dogs, a total of 130 breeds collected (Fig. 4).

4.1.5. News-20. The 20 Newsgroups dataset [28] is usually used to compare classification methods, but in

Table 1. Partition into training and test sets, classes between partitions do not intersect

	Number of classes in the training set	Number of classes in the test set	Number of objects in the training set	Number of objects in the test set
Cars-196	98	98	8006	8179
CUB-200	100	100	5897	5891
SOP	11 317	11 317	59 989	60 064
Dogs-130	65	65	36 904	33 528
News-20	10	10	9622	9224
WOS-134	67	67	24 283	22 702

this work we use it to build textual representations. In total, the dataset contains 18 846 texts and 20 classes.

4.1.6. WOS-134. The Web Of Science dataset [29] contains a total of 46 985 text documents. The number of categories is quite big, viz. 134, so Deep Metric Learning approaches may be especially relevant there.

4.2. Neural Network Architectures

4.2.1. ConvNeXt. Most of the previously discussed methods in the original articles were compared on architectures from 2014–2015, viz. Resnet-50 [30] and GoogLeNet [31]. Since then, many other models have emerged for which metric learning has not been explored. In this work, ConvNeXt is used as the principal neural network for images [32], which has a convolutional architecture yet it is comparable in quality to modern transformer models. We use pretrained ConvNeXt-T, which has 28M trainable parameters. On top of the neural network, we add a linear layer that maps the hidden state into the space of the required dimension.

4.2.2. DistilBERT. To work with text data, we use the DistilBERT transformer model [33], which is the distilled version of BERT [34]. To obtain vector representations of the required dimension, we also add a linear layer on top of the model, specifically the CLS token.

4.2.3. Training details. PyTorch and [35] were the principal frameworks used for writing training and measurement scripts. Adam [36] was used as an optimizer for all models, the learning rate varied from 10^{-5} to 10^{-3}, the batch size varied from 32 to 128 (the best quality for each model is given below). The hyperparameters of the functions themselves are fixed in accordance with the authors' results. Optimization used early stopping if the quality did not improve within 10 epochs. In the case of DistilBERT, warmup was additionally used for the first 100 iterations. For images during training, standard augmentation methods are used: random horizontal reflection, clipping and scaling of a random segment of the image. When measuring quality, test images are scaled to 256, and then the central 224×224 fragment is cropped.

4.3. Results

In this work, all loss functions described in Section 3 are compared on the problems discussed above, viz. those based on image recognition (Cars-196, CUB-200, SOP, Dogs-130) and natural language (News-20, WOS-134). Note that many methods, individually or in small sets, were tested on the Cars-196, CUB-200, and SOP datasets in the works where the methods were first proposed, with the author's algorithms often achieving the highest quality. Further, we describe the results of comparing Deep Metric Learning approaches under completely equal conditions.

On Cars-196 (Table 2) with ConvNeXT, Tuplet Margin Loss worked best, but the highest values of MAP functionals were achieved when using Circle Loss (9). Indeed, from the point of different quality criteria, different methods may be better or worse (often only Recall@K is given in articles). Note that Tuplet Loss also worked better for MRR, NMI, and AMI on Cars-196.

We consider News-20 (20 Newsgroups), which is a text dataset with 20 classes, and a transformer model—Table 3 shows that Tuplet Margin Loss and Triplet

Fig. 1. Cars-196.

Fig. 2. CUB-200: random test subset after preprocessing.

Fig. 3. SOP: random test subset after preprocessing.

Fig. 4. Dogs-130: random test subset after preprocessing.

Loss optimize better on Recall while MAP functionals are better when we use Multi Similarity Loss, and cluster functionals are better when we use Circle Loss.

On CUB-200 (Table 4), the results largely repeat Cars-196, viz. Tuplet Loss and Circle Loss are of the highest quality.

Table 2. Evaluation metrics for Cars-196 test

	R@1	R@2	R@4	R@8	R@16	R@32	MAP	MAP@R	MRR	AMI	NMI
Contrastive Loss	86.14	91.80	95.15	97.15	98.32	99.17	47.23	33.05	90.49	70.59	74.69
Triplet Loss	85.68	91.80	95.61	97.42	98.41	99.11	49.21	35.05	90.34	73.39	77.10
Fast AP	83.36	90.01	93.95	96.38	97.79	98.78	48.92	34.41	88.49	70.47	74.60
Centroid Triplet Loss	83.80	90.82	95.08	7.63	98.90	**99.50**	40.99	27.04	89.17	70.38	74.48
Margin Loss	81.82	88.12	92.69	95.33	97.14	98.21	44.16	29.94	87.03	65.46	70.24
Multi Similarity Loss	86.89	92.58	95.62	97.58	98.74	99.19	50.99	36.74	91.12	73.84	77.49
SNN Loss	84.36	90.38	94.11	96.65	97.92	98.85	49.57	35.35	89.10	72.29	76.16
SupCon Loss	81.01	88.32	92.73	95.59	97.32	98.42	46.35	32.28	86.70	68.70	73.06
SNR Loss	86.88	92.24	95.27	97.31	98.61	99.28	48.41	34.00	91.00	70.11	74.27
Tuplet Margin Loss	**88.54**	**94.11**	**96.88**	**98.37**	**99.11**	99.49	50.53	36.24	**92.51**	**76.10**	**79.41**
Circle Loss	88.05	92.92	95.95	97.76	98.63	99.30	**52.81**	**38.46**	91.85	75.00	78.48
ArcFace	87.35	92.69	95.45	97.26	98.35	99.18	50.74	35.82	91.30	71.56	75.52
CosFace	87.06	92.32	95.23	97.21	98.34	99.08	50.94	35.72	91.06	72.84	76.62
SubCenter ArcFace	87.13	92.68	95.57	97.48	98.67	99.23	51.26	36.35	91.24	72.00	75.89
SoftTriple Loss	86.76	92.63	95.90	97.68	98.75	99.39	48.14	33.71	91.13	72.00	75.91
Proxy-Anchor Loss	88.43	93.48	96.06	97.86	98.88	99.43	52.30	36.34	92.16	74.42	77.92

Table 3. Evaluation metrics for 20 Newsgroups test

	R@1	R@2	R@4	R@8	R@16	R@32	MAP	MAP@R	MRR	AMI	NMI
Contrastive Loss	77.95	84.92	89.80	93.83	97.09	98.45	56.99	32.92	83.93	50.15	50.25
Triplet Loss	78.48	**85.41**	**90.44**	94.36	97.35	98.83	58.64	34.66	84.46	50.49	50.58
Fast AP	77.07	84.50	89.59	93.76	97.21	98.68	57.75	33.78	83.41	50.34	50.44
Centroid Triplet Loss	78.56	85.26	90.03	94.02	97.21	98.69	58.38	33.97	84.38	49.62	49.72
Margin Loss	77.16	84.72	89.72	93.85	97.25	98.54	58.04	34.03	83.52	49.46	49.56
Multi Similarity Loss	78.25	84.92	89.68	93.71	97.15	98.66	**59.01**	**35.40**	84.08	52.60	52.69
SNN Loss	77.70	84.37	89.48	93.82	97.08	98.54	58.34	35.00	83.67	50.78	50.88
SupCon Loss	77.44	84.56	89.68	93.92	97.30	98.67	57.92	34.19	83.63	50.52	50.61
SNR Loss	77.86	84.57	89.67	93.78	97.13	98.42	58.37	34.53	83.82	51.36	51.45
Tuplet Margin Loss	**78.65**	85.26	90.18	94.56	97.60	98.92	57.79	33.39	**84.51**	52.98	53.07
Circle Loss	78.53	85.23	90.34	**94.63**	**98.01**	**99.17**	56.84	32.24	84.50	**53.39**	**53.48**
ArcFace	76.24	83.95	89.05	93.87	97.25	98.79	56.56	33.32	82.84	49.97	50.06
CosFace	76.40	83.67	88.99	93.56	96.98	98.66	57.02	33.92	82.82	49.71	49.81
SubCenter ArcFace	76.51	83.79	89.09	93.27	96.76	98.31	57.61	34.85	82.82	49.97	50.07
SoftTriple Loss	77.02	83.86	89.26	93.61	97.13	98.75	57.22	33.86	83.18	48.95	49.05
Proxy-Anchor Loss	76.98	83.88	89.68	93.73	97.20	98.59	56.96	33.63	83.24	51.35	51.45

Table 4. Evaluation metrics for CUB-200 test

	R@1	R@2	R@4	R@8	R@16	R@32	MAP	MAP@R	MRR	AMI	NMI
Contrastive Loss	81.46	88.08	92.85	95.88	97.78	98.83	53.67	40.96	86.95	77.66	81.80
Triplet Loss	82.69	89.34	93.62	96.37	98.13	98.88	56.14	43.18	87.97	78.61	82.51
Fast AP	80.94	87.57	92.43	95.13	97.10	98.27	55.00	41.82	86.44	76.63	80.91
Centroid Triplet Loss	79.48	87.71	92.85	96.16	97.96	98.73	48.77	36.26	85.92	76.09	80.45
Margin Loss	77.36	85.13	90.53	94.72	96.83	98.05	49.53	36.50	83.84	70.00	75.46
Multi Similarity Loss	82.07	88.76	92.97	96.03	97.62	98.46	55.84	42.98	87.41	77.00	81.24
SNN Loss	82.01	88.68	93.21	96.06	97.67	98.57	56.64	43.58	87.39	77.97	82.02
SupCon Loss	80.19	87.52	92.34	95.67	97.39	98.42	55.03	41.82	86.10	76.43	80.77
SNR Loss	82.28	88.80	93.50	96.15	98.05	98.90	55.01	42.34	87.62	77.80	81.92
Tuplet Margin Loss	**83.36**	89.81	**93.96**	**96.89**	**98.18**	**99.02**	55.66	42.68	**88.54**	**79.88**	**83.55**
Circle Loss	83.35	**89.93**	93.74	96.28	97.93	98.83	**57.68**	**44.78**	88.44	79.29	83.13
ArcFace	80.61	87.22	91.78	94.89	97.06	98.40	54.70	41.44	86.11	73.91	78.79
CosFace	80.75	87.27	91.73	94.82	97.00	98.25	53.80	40.63	86.18	75.10	79.74
SubCenter ArcFace	80.12	87.15	92.04	94.87	96.77	98.00	54.72	41.57	85.83	73.87	78.69
SoftTriple Loss	80.82	87.74	92.43	95.76	97.66	98.69	53.48	40.50	86.49	77.07	81.32
Proxy-Anchor Loss	80.82	88.02	92.36	95.65	97.45	98.51	53.97	40.81	86.52	76.18	80.55

Dataset Dogs-130 (Table 5) is quite simple to optimize as its classes contain a particularly large number of examples. Nevertheless, it was clearly better to use Tuplet Loss on it, both from the point of Recall and AMI / NMI. MAP turned out to be the best for SNN Loss.

On SOP (*Table 6: the largest number of classes considered in this dataset*), Circle Loss is the most optimal one in terms of Recall and MAP functionals, although Multi Similarity Loss performed better on AMI/NMI.

On the final WOS-134 text dataset, SNR Loss has the highest MAP (Table 7) while cluster metrics are better for Centroid Triplet Loss. At the same time, Tuplet Margin Loss again turned out to be the best on Recall@1.

The results obtained for image datasets can be considered as new benchmarks for Deep Metric Learning that are based on ConvNeXT rather than on Resnet or GoogleNet. Based on the results of experiments, the methods often have a higher quality than in the origi-

Table 5. Evaluation metrics for Dogs-130 test

	R@1	R@2	R@4	R@8	R@16	R@32	MAP	MAP@R	MRR	AMI	NMI
Contrastive Loss	94.23	96.72	97.84	98.54	99.01	99.31	85.95	76.42	95.99	77.40	77.80
Triplet Loss	94.47	96.88	97.98	98.64	99.08	99.40	86.36	77.05	96.18	78.21	78.60
Fast AP	94.25	96.74	97.88	98.55	98.97	99.32	86.54	77.46	96.01	78.00	78.39
Centroid Triplet Loss	**94.57**	96.89	98.04	**98.74**	99.13	99.46	84.69	73.79	**96.25**	79.14	79.51
Margin Loss	94.30	96.78	97.86	98.59	99.00	99.29	86.47	77.36	96.05	77.98	78.38
Multi Similarity Loss	94.23	96.72	97.84	98.54	99.01	99.31	85.95	76.42	95.99	77.40	77.80
SNN Loss	94.28	96.68	97.89	98.55	98.97	99.28	**86.77**	**77.92**	96.02	77.89	78.29
SupCon Loss	94.27	96.75	97.90	98.58	99.00	99.36	86.18	76.84	96.04	77.66	78.06
SNR Loss	94.23	96.72	97.84	98.54	99.01	99.31	85.95	76.42	95.99	77.40	77.80
Tuplet Margin Loss	94.49	**96.92**	**98.09**	98.71	**99.16**	**99.46**	85.26	74.87	96.23	**79.43**	**79.81**
Circle Loss	94.36	96.84	97.88	98.55	98.97	99.31	86.40	77.27	96.09	77.80	78.20
ArcFace	94.29	96.69	97.83	98.51	98.99	99.32	85.90	76.38	96.01	77.75	78.15
CosFace	94.29	96.66	97.82	98.52	98.97	99.31	85.86	76.34	96.01	77.82	78.21
SubCenter ArcFace	94.27	96.75	97.87	98.52	99.02	99.32	85.99	76.48	96.02	77.47	77.87
SoftTriple Loss	94.25	96.67	97.80	98.53	98.99	99.31	86.06	76.80	95.99	77.96	78.36
Proxy-Anchor Loss	94.31	96.73	97.85	98.54	98.97	99.32	86.15	76.81	96.04	78.00	78.39

Table 6. Evaluation metrics for SOP test

	R@1	R@10	R@100	R@1000	MAP	MAP@R	MRR	AMI	NMI
Contrastive Loss	81.08	90.60	95.91	98.70	63.56	54.87	84.45	54.14	90.87
Triplet Loss	80.31	91.61	96.92	99.20	61.69	53.02	84.30	50.82	90.16
Fast AP	80.80	91.69	96.89	99.17	63.51	54.94	84.63	54.19	90.83
Centroid Triplet Loss	76.55	89.22	95.83	98.88	56.33	47.22	81.01	45.90	89.11
Margin Loss	73.09	86.52	94.82	98.70	52.58	44.16	77.81	44.74	88.85
Multi Similarity Loss	81.32	91.30	96.56	98.99	64.05	55.71	84.83	**55.61**	**91.15**
SNN Loss	80.58	91.66	96.79	99.23	62.80	54.27	84.47	53.71	90.73
SupCon Loss	75.92	88.45	95.44	98.83	56.65	48.05	80.29	49.30	89.84
SNR Loss	81.98	91.55	96.38	98.82	64.98	56.28	85.36	54.24	90.91
Tuplet Margin Loss	50.72	66.17	79.99	92.71	34.63	22.88	56.11	24.10	84.29
Circle Loss	**82.14**	**92.50**	**97.08**	**99.25**	**65.27**	**56.87**	**85.80**	55.45	91.09
ArcFace	64.18	75.92	82.94	89.83	49.70	34.17	68.36	30.57	86.24
CosFace	64.17	75.64	82.94	89.91	49.27	33.89	68.21	30.10	86.17
SubCenter ArcFace	61.58	73.10	81.08	88.83	47.24	31.53	65.66	27.60	85.63
SoftTriple Loss	81.37	91.44	96.12	98.48	64.15	54.49	84.94	51.66	90.39
Proxy-Anchor Loss	80.68	91.48	96.33	98.60	63.55	53.63	84.52	51.68	90.35

nal articles, for example, on the SOP dataset with ConvNeXT, the obtained Recal@K are higher than in [9, 12, 13, 18, 22, 23]. For the text datasets used here, such statement of the problem was previously considered in [37], but the approach using a transformer model and loss functions from computer vision is probably being tested for the first time.

5. CONCLUSIONS

We give an overview of loss functions that were previously proposed for optimizing neural networks in deep metric learning problems. Experiments were carried out “under equal conditions” with the described loss functions for different domains, viz. images and texts, as well as with modern neural network architec-

Table 7. Evaluation metrics for WOS-134 test

	R@1	R@2	R@4	R@8	R@16	R@32	MAP	MAP@R	MRR	AMI	NMI
Contrastive Loss	57.02	68.03	77.12	84.06	89.16	93.01	35.81	18.46	67.15	50.59	51.80
Triplet Loss	58.40	70.47	**79.88**	86.28	91.15	94.65	34.36	16.48	69.00	50.13	51.35
Fast AP	55.91	67.92	77.27	84.40	89.44	93.13	34.29	16.96	66.61	49.17	50.42
Centroid Triplet Loss	59.06	**70.76**	79.68	86.53	91.52	94.80	34.51	16.73	**69.40**	**51.81**	**52.99**
Margin Loss	57.22	68.74	78.02	85.22	90.15	93.79	33.10	15.43	67.65	48.61	49.88
Multi Similarity Loss	57.72	69.05	78.02	84.89	90.15	93.69	35.76	18.48	67.92	50.92	52.13
SNN Loss	57.96	69.44	78.19	85.09	90.10	93.72	35.56	17.96	68.16	50.19	51.41
SupCon Loss	57.91	69.76	79.02	85.85	90.69	94.25	34.63	16.99	68.43	50.23	51.45
SNR Loss	57.33	68.28	77.05	83.80	88.91	92.96	**35.87**	**18.67**	67.30	49.84	51.08
Tuplet Margin Loss	**59.08**	70.33	79.33	85.84	91.02	94.40	35.13	17.33	69.20	50.29	51.51
Circle Loss	58.50	70.10	79.61	**86.58**	**91.58**	**94.86**	34.06	16.22	68.97	51.43	52.62
ArcFace	54.00	66.36	76.20	83.90	89.47	93.60	30.61	13.59	65.17	45.45	46.80
CosFace	54.65	66.37	75.83	83.25	88.64	92.68	32.34	15.30	65.35	46.73	48.04
SubCenter ArcFace	55.85	67.40	76.90	84.00	89.34	93.12	33.65	16.49	66.40	48.95	50.21
SoftTriple Loss	55.90	67.48	76.70	84.05	89.30	93.05	32.29	15.07	66.40	47.21	48.51
Proxy-Anchor Loss	56.62	68.08	77.60	84.82	90.23	93.64	32.63	15.25	67.11	47.96	49.24

tures. As expected, the choice of a loss function depends on the problem, model and evaluation metrics. Nevertheless, in most of the experiments conducted, Tuplet Margin Loss and Circle Loss were more likely to achieve the best results than other methods, which is a little surprising considering that the first one was proposed in 2019 and there are many "more recent" methods.

We also note that we sometimes managed to achieve better quality in our experiments than in the works in which the loss functions involved were proposed. Although it is possible to perform a sufficient number of experiments with different partitions of datasets into training and test with the construction of confidence intervals for the results obtained, we did not do it due to the complexity of such a series of experiments. Among the interesting directions in which the work can be developed, we should mention the case when the labels take real values. Basically, loss functions are focused on categorical label values, so the analytical record contains objects with equal and unequal labels. Intuitively, in the case of real labels, the closer the labels, the closer the representations should be, but various formalizations of this idea have not yet been fully explored.

FUNDING

This work was supported by the ongoing institutional funding. No additional grants to carry out or direct this particular research were obtained.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. W. Chen, Y. Liu, W. Wang, E. M. Bakker, T. K. Georgiou, P. Fieguth, L. Liu, and M. S. K. Lew, "Deep image retrieval: A survey" (2021). https://arxiv.org/abs/2101.11282
2. N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using Siamese bert-networks" (2019). arXiv:1908.10084
3. I. Masi, Y. Wu, T. Hassner, and P. Natarajan, "Deep face recognition: A survey," in *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images* (IEEE, 2018), pp. 471–478.
4. M. Je, J. Shen, G. Lin, T. Xiang, L. Shao, and C. H. S. Hoi, "Deep learning for person re-identification: A survey and outlook," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).
5. K. Musgrave, S. Belongie, and S.-N. Lim, "A metric learning reality check," in *European Conference on Computer Vision* (Springer, Berlin, 2020), pp. 681–699.
6. J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," IEEE Trans. Big Data **7** (3), 535–547 (2019).
7. S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005), Vol. 1, pp. 539–546.
8. F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and cluster-

ing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 815–823.

9. F. Cakir, K. He, X. Xia, B. Kulis, and S. Sclaroff, "Deep metric learning to rank," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 1861–1870.
10. E. Ustinova and V. Lempitsky, "Learning deep embeddings with histogram loss," *Advances in Neural Information Processing Systems* (2016), Vol. 29.
11. V. Wieczorek, B. Rychalska, and J. Dąbrowski, "On the unreasonable effectiveness of centroids in image retrieval," in *International Conference on Neural Information Processing* (Springer, Berlin, 2021), pp. 212–223.
12. W. Chao-Yuan, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2840–2848.
13. W. Xun, H. Xintong, H. Weilin, D. Dengke, and M. R. Scott, "Multisimilarity loss with general pair weighting for deep metric learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 5022–5030.
14. N. Frosst, N. Papernot, and G. Hinton, "Analyzing and improving representations with the soft nearest neighbor loss," Proc. Mach. Learn. Res. **97**, 2012–2020 (2019).
15. P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," Adv. Neural Inf. Process. Syst. **33**, 18661–18673 (2020).
16. T. Yuan, W. Deng, J. Tang, Y. Tang, and B. Chen, "Signal-to-noise ratio: A robust distance metric for deep metric learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 4815–4824.
17. B. Yu and D. Tao, "Deep metric learning with tuplet margin loss," in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 6490–6499.
18. Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei, "Circle loss: A unified perspective of pair similarity optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 6398–6407.
19. J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 4690–4699.
20. H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 5265–5274.
21. J. Deng, J. Guo, T. Liu, M. Gong, and S. Zafeiriou, "Subcenter arcface: Boosting face recognition by large-scale noisy web faces," in *European Conference on Computer Vision* (Springer, Berlin, 2020), pp. 741–757.
22. Q. Qian, L. Shang, B. Sun, J. Hu, H. Li, and R. Jin, "Softtriple loss: Deep metric learning without triplet sampling," in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 6450–6458.
23. S. Kim, D. Kim, M. Cho, and S. Kwak, "Proxy anchor loss for deep metric learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 3238–3247.
24. J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D object representations for fine-grained categorization," in *The 4th International IEEE Workshop on 3D Representation and Recognition, Sydney, Australia* (2013).
25. C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, Caltech-UCSD Birds-200-2011 Dataset (2011).
26. H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*) (2016).
27. D.-N. Zou, S.-H. Zhang, T.-J. Mu, and M. Zhang, "A new dataset of dog breed images and a benchmark for fine-grained classification," Comput. Visual Media **6** (4), 477–487 (2020).
28. K. Lang, "Newsweeder: Learning to filter netnews," in *Machine Learning Proceedings* (Elsevier, 1995), pp. 331–339.
29. K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," in *2017 16th IEEE International Conference on Machine Learning and Applications* (*ICMLA*) (IEEE, 2017), pp. 364–371.
30. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.
31. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015). pp. 1–9.
32. Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xiell, "A ConvNet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*) (2022).
33. V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter" (2019). https://arxiv.org/abs/1910.01108
34. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding" (2018). https://arxiv.org/abs/1810.04805
35. K. Musgrave, S. Belongie, and S.-N. Lim, "PyTorch metric learning" (2020). https://arxiv.org/abs/2008.09164
36. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization" (2014). https://arxiv.org/abs/1412.6980
37. J. Wohlwend, E. R. Elenberg, S. Altschul, S. Henry, and T. Lei, "Metric learning for dynamic text classification" (2019). https://arxiv.org/abs/1911.01026