Similarity Metric Learning

Stefan Duffner, Christophe Garcia, Khalid Idrissi, Atilla Baskurt

Similarity Metric Learning

Stefan Duffner, Christophe Garcia, Khalid Idrissi and Atilla Baskurt

Abstract Similarity metric learning models the general semantic similarities and distances between objects and classes of objects (*e.g.* persons) in order to recognise them. Different strategies and models based on Deep Learning exist and generally consist in learning a non-linear projection into a lower dimensional vector space where the semantic similarity between instances can be easily measured with a standard distance. As opposed to supervised learning, one does not train the model to predict the class labels, and the actual labels may not even be used or not known in advance. Machine learning-based similarity metric learning approaches rather operate in a weakly supervised way. That is, the training target (loss) is defined on the relationship between several instances, *i.e.* similar or different pairs, triplets or tuples. This learnt distance can then be applied, for example, to two new, unseen examples of unknown classes in order to determine if they belong to the same class or if they are similar. There exist numerous applications for metric learning such as face or speaker verification, image retrieval, human activity recognition or person re-identification in images. In this chapter, an overview of the principle methods and models used for similarity metric learning with neural networks is given, describing the most common architectures, loss functions and training algorithms.

1 Introduction

Traditionally, neural networks, and in particular deep neural networks, have been used in settings involving supervised learning, and they showed state-of-the-art performance in many applications where abundant annotated data are available. In these applications, it is usually required to know in advance

e-mail:

the number of classes to predict, and their clear meaning, *i.e.* which instance belongs to them or not, or, in case of a regression problem: the values corresponding to the instances. Moreover, the relationship of different classes (or sub-classes) is not explicitly modelled, as this is mostly not useful in the given predictive scenarios. In this part, we will consider other applications that do not allow or are not suited for such supervised learning approaches. This is the case, for example, when:

- instance labels (*e.g.* positive/negative, foreground/background or a person identifier) are not available or too difficult to obtain for training or
- the number of classes is not fixed *a priori* or
- the classes to predict at test time are not the same as the ones available for training or
- the relationship or similarity between instances and categories of instances should be modelled and represented explicitly or
- a robust rejection strategy needs to be implemented for a classification task.

If no information on instance classes or their relationship is given at all, *unsupervised approaches*, such as clustering methods or auto-encoders, are most suitable to automatically learn and infer a general model of the data. However, in many settings, class labels for at least some of the training data is available, and one is interested in a generic model that is applicable for all data of the same type. In this case, *weakly supervised* or *semi-supervised learning* algorithms are commonly employed.

One such weakly supervised approach is to automatically learn a similarity metric between instances of a given category (*e.g.* faces). That is, the instance labels of a training dataset are not explicitly learnt but rather used to model the distance between similar and dissimilar instances – for example, by using pairs of instances. Note that sometimes the term *distance learning* is used. But many existing models, *e.g.* those based on neural networks described in this chapter, do not fulfil the mathematical requirements of a distance, especially the triangle inequality, and thus do not represent a metric neither.

The notion of similarity or dissimilarity differs from one application to another. Thus its definition depends on the learning problem and on the training dataset. For example, let us consider a set of face images: (1) for the task of face verification, two images of the same person are considered similar; (2) for the task of gender verification, two face images showing two males or two females are defined as similar; (3) for the task of kinship verification, a similar pair of face images must indicate a biological relationship such as father-son, mother-daughter *etc.*

In principle, most metric learning algorithms receive positive and negative pairs of instances, where a positive pair is formed of instances that are considered similar (*e.g.* belonging to the same category), and a negative pair is composed of instances that are considered different or dissimilar, although this may not be simple to determine in practice. By presenting all possible

pairs, a good metric learning method must then be able to capture the intrinsic relationship between the concerned semantic contents of two objects. Figure 1 shows pairs of images of the same persons from the LFW dataset. These pairs can be used to define a similarity relationship, and a generic dis-

Fig. 1 Pairs of images (LFW dataset) from the same person that can be used to learn a similarity relationship. Similarity metric learning aims at learning a metric space where similar examples (*e.g.* faces from the same person) are close and examples not considered similar have a large distance. This metric can then be used to verify if two unknown examples, possibly from unknown classes, belong to the same class (*e.g.* for face verification).

tance metric can be learnt and applied to unknown persons, for example for face verification. An alternative approach is to give triplets of instances, with one reference (also called "anchor"), one similar instance and one dissimilar instance w.r.t. to the reference. Then, the similar instance is considered more similar to the reference than the third one. Or, sometimes quadruplets are used, where the first pair is considered more similar than the second pair. Figure 2 illustrates the principle approach for similarity metric learning with pairs or triplets of instances.

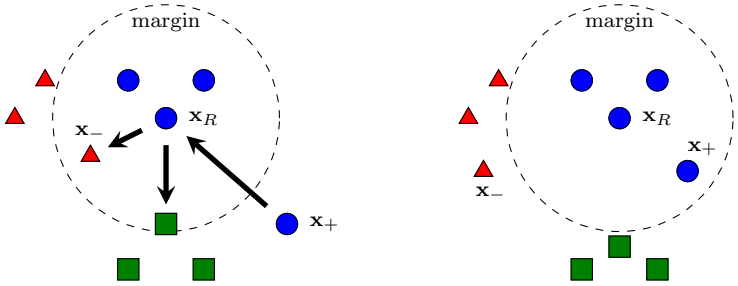

Fig. 2 Principle approach for similarity metric learning. Minimising an appropriate loss function tries to bring similar instances closer $(\mathbf{x}_R, \mathbf{x}_+)$ and to separate dissimilar ones $(\mathbf{x}_R, \mathbf{x}_-)$ (if they are closer than a certain margin). A *pair-wise* loss function iteratively optimises these objectives on all positive and negative pairs, whereas a *triplet*-based loss is defined on triplets $(\mathbf{x}_R, \mathbf{x}_+, \mathbf{x}_-)$ (*c.f.* Section 2.3).

In this chapter, we will present similarity metric learning methods and models based on Siamese Neural Networks (SNN). The original term "siamese" relates to the use of pairs of instances as introduced by Bromley *et al.* [BGL+94]. However, in the literature, this has been extended to triplet or tuple-based architectures. As we will outline in the next section, there are other models, for example, based on statistical projections or Support Vector Machines. However, feed-forward neural networks have several properties that make them interesting and particularly suitable for this problem:

- They can model a wide variety of linear and non-linear functions, *c.f.* the universal approximation theorem [HSW89], and well-established optimisation approaches can be used.
- By carefully specifying the architectures, the complexity of the models can be controlled (in terms of the number of parameters and the dimensions at different abstraction levels, considering it as a data processing pipeline).
- Through multi-layered architectures and the error back-propagation algorithm, one can jointly learn optimal features and projections into vector spaces that best represent semantic similarities.
- Using deep Convolutional Neural Network architectures, the resulting models are suitable and very powerful for (natural) image data.
- Finally, for a large variety of applications and data, neural networks showed a high generalisation capacity and robustness to different types of noise.

In the following, we will outline the different algorithms and neural models that exist for similarity metric learning and describe the principal SNN approach and variants that have been presented in the literature.

2 Metric learning with Neural Networks

Most linear metric learning methods employ two types of metrics: the Mahalanobis distance or a more general similarity metric. In both cases, a linear transformation matrix W is learnt projecting input features into a target vector space. Typically, distance metric learning relates to a Mahalanobis-like distance function [XNJR03, WBS06]: $d_W(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T W (\mathbf{x}_1 - \mathbf{x}_2)}$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are two example vectors, and W is not the (fixed) covariance matrix, as for the Mahalanobis distance, but is to be learnt by the algorithm. Note that when W is the identity matrix, $d_W(\mathbf{x}_1, \mathbf{x}_2)$ corresponds to the Euclidean distance. In contrast, similarity metric learning methods learn a function of a more general form: $s_W(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T W \mathbf{x}_2 / N(\mathbf{x}_1, \mathbf{x}_2)$, where $N(\mathbf{x}_1, \mathbf{x}_2)$ is a normalisation term [QGCL08]. Specifically, when $N(\mathbf{x}_1, \mathbf{x}_2) = 1$, $s_W(\mathbf{x}_1, \mathbf{x}_2)$ is the bilinear similarity function [CSSB10]; and when $N(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\mathbf{x}_1^T W \mathbf{x}_1} \sqrt{\mathbf{x}_2^T W \mathbf{x}_2}$, $s_W(\mathbf{x}_1, \mathbf{x}_2)$ corresponds to the generalised cosine similarity function [HL11].

Non-linear metric learning methods are constructed by simply substituting the above linear projection with a non-linear transformation [HLT14, CHL05, KTS+12, YYGT16]. For example, Hu *et al.* [HLT14] and Chopra *et al.* [CHL05] employed neural networks to accomplish this. With these approaches, the learning algorithm optimises the parameters of a non-linear projection $\mathbf{o} = f(\mathbf{x})$ into a metric space such that a simple distance measure $d_W(f(\mathbf{x}_1), f(\mathbf{x}_2))$, mostly the Euclidean or cosine distance, reflects the semantic similarities between the instances. Such non-linear methods are subject to local optima and more inclined to overfit the training data but have the potential to outperform linear methods on many problems [BHS13, KTS+12]. Compared with linear models, non-linear models are usually preferred on large training sets to more accurately capture the underlying distribution of the data [LBOM12]. A detailed survey and review of metric learning approaches has been published recently by Bellet *et al.* [BHS13], and an experimental analysis and comparison by Moutafis *et al.* [MLK16]. We will concentrate here on Siamese Neural Networks (SNN) that can represent linear or non-linear projections depending on the used activation function and the number of layers. With larger and more complex models, the term *deep similarity metric learning* has often been employed in the literature.

As mentioned before, a SNN essentially distinguishes itself from classical feed-forward neural networks by its specific training strategy involving sets of examples labelled as similar or dissimilar. The capabilities of different SNN-based methods depend on four main points: the network architecture, the training set selection strategy, the loss function, and the training algorithm [BGL+94]. In the following, we will explain these points in more detail and give some examples.

2.1 Architectures

A SNN can be seen as two identical, parallel neural networks *NN* sharing the same set of weights W (see Fig. 3). Although the weights of the input branches are shared in most cases, this may not be the case for some specific applications, notably when the two input examples are of different type or from different sources (*e.g.* images taken from two different camera views). Then, each input branch of the neural network is dedicated to a specific input type, and the two cannot be interchanged. In any case, the sub-networks receive input examples $\mathbf{x}_1, \mathbf{x}_2$, and produce output feature vectors $\mathbf{o}_1, \mathbf{o}_2$ that are supposed to be *close* for examples from the same class and *far apart* for examples from different classes, according to some distance measure, such as the cosine similarity metric (*c.f.* section 2.3). During the training phase, a loss function $\mathcal{L}_W$, defined using the chosen distance measure over the output of all input examples, is iteratively minimised (*c.f.* Section 2.3). The architecture of the neural network determines the type and complexity of the projection

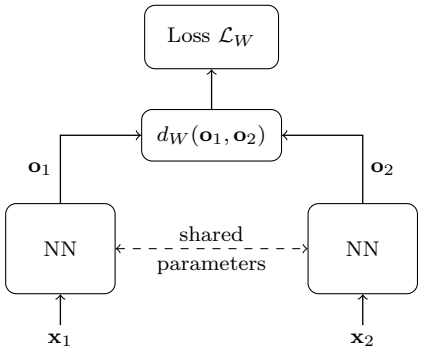

Fig. 3 The principal SNN architecture. Two identical neural networks NN receive two different input vectors $\mathbf{x}_1, \mathbf{x}_2$ producing two embeddings $\mathbf{o}_1$ and $\mathbf{o}_2$, respectively. The loss function $\mathcal{L}_W$ to minimise is based on a distance measure d_W between these embeddings and is used to update the shared weights of the neural network computing the gradient w.r.t. to both of them and back-propagating the error.

function. One crucial parameter is the size of the output layer as it defines the dimension of the vector space of the embedding. This is usually an empirical choice. It should be large enough to capture the similarity relationships of the training data (according to different aspects/axes) but not too large as the distance measures in too large vector spaces may become extremely small and thus meaningless.

Bromley *et al.* [BGL+94] introduced the Siamese architecture in 1994, using a Siamese CNN with two sub-networks for a signature verification system handling time-series of hand-crafted low-level features. In 2005, Chopra, Hadsell and LeCun [CHL05] formalised the Siamese architecture applying a CNN on raw images for face verification, before adapting it to a dimensionality reduction technique [HCL06] (see Fig. 4(b) for an illustration of a siamese CNN architecture). More recently, Siamese CNNs have been used successfully for various tasks, such as person re-identification [YLLL14], speaker verification [CS11], and face identification [SCWT14].

CNN-based architectures are more specific to image inputs, and several research works propose to use feed-forward Multi-Layer Perceptrons (MLP) to handle more general vector inputs (*c.f.* Fig. 4(a)). For example, Yih *et al.* [YTPM11] apply SNNs to learn similarities on text data, Bordes *et al.* [BWCB11] on entities in Knowledge Bases, and Masci *et al.* [MBBS14] on multi-modal data. Recently, siamese or triplet architectures have become popular and have been used for many different applications and types of data and combined with other types of neural models. For example, for sequence or time series modelling with recurrent neural networks as illustrated in Fig. 4(c). Mueller and Thyagarajan [MT16] proposed a Siamese neural network for learning similarities between sentences using Long Short-Term Memory (LSTM) neural networks, where the distances are computed on the last

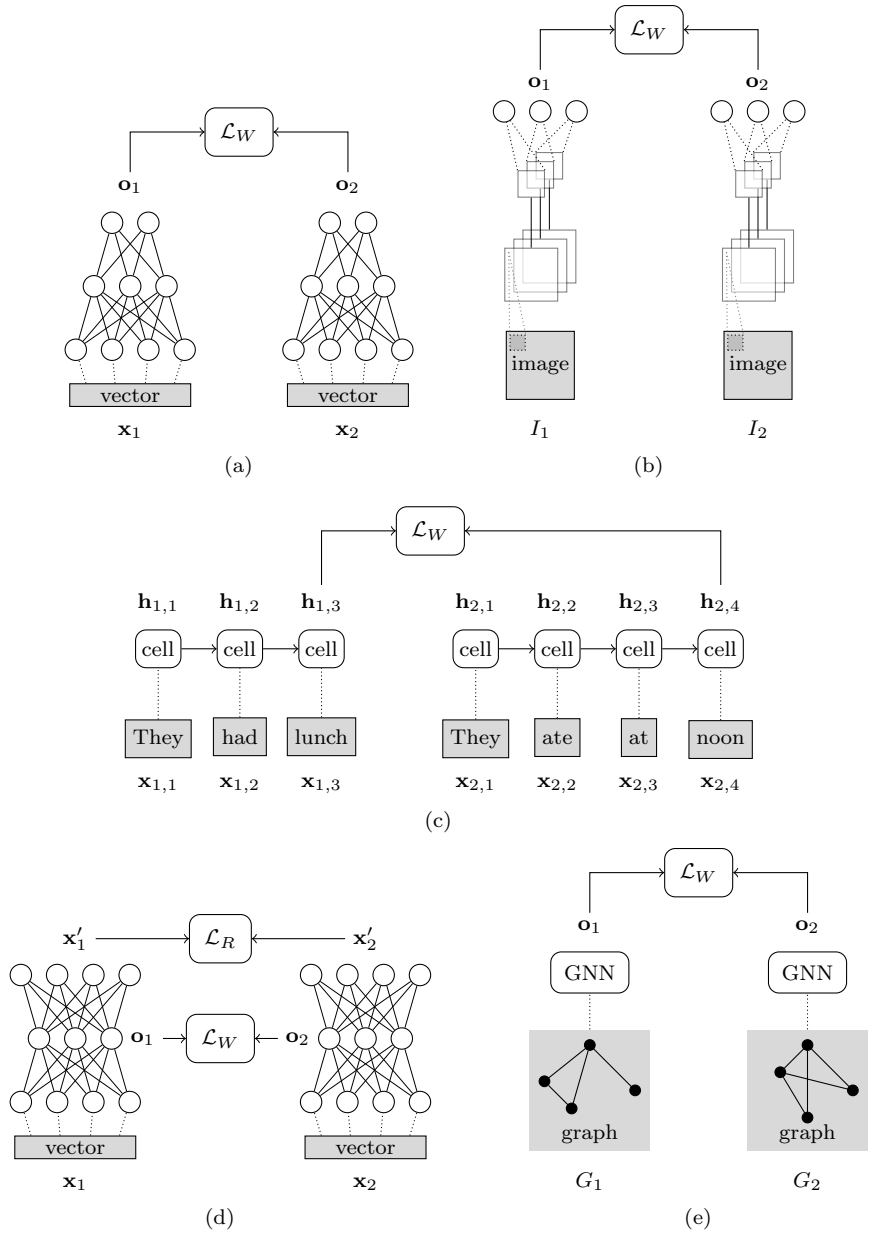

Fig. 4 Examples of different types of neural network models for deep similarity metric learning where two inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ ($I_1, I_2, G_1, G2$, respectively) produce fixed-size output embeddings $\mathbf{o}_1$ and $\mathbf{o}_2$ that are used for minimising the loss function $\mathcal{L}_W$: a) Multi-Layer Perceptrons (MLP) for vector data, b) Convolutional Neural Network (CNN) for images, c) Recurrent Neural Networks for variable-length sequential data (here: vectors of word embeddings), d) auto-encoders minimising simultaneously the reconstruction loss $\mathcal{L}_R$ and e) Graph Neural Networks operating on graph structures. For each of the architectures, only two branches are shown, but these models can be easily adapted to several inputs (triplets, quadruplets *etc.*).

hidden layer output. Similarly, the approach from Neculoiu *et al.* [NVR16] uses a Bidirectional LSTM to compute similarities between words. Here, the hidden layer outputs are averaged over the sequence in order to obtain two embedding vectors of the same dimension. SNNs have also been used with auto-encoder architectures [KBR16, YCS19, SL20, ASS20] (*c.f.* Fig. 4(d)). In that case, the two (or more) auto-encoders share their weights and the siamese or triplet loss is minimsed on the latent embedding (code) produced by the encoder, whereas the reconstruction loss is optimised at the same time. This approach has been extended by Compagnon *et al.* [CLDG19] for sequence metric learning with a LSTM-based sequence-to-sequence auto-encoder. Such multi-task learning structures and algorithms have very popular in the past years and successfully applied to various domains, notably Computer Vision. Recently, Graph Neural Networks (GNN) have been combined with pairwise or triplet-based similarity metric learning for image retrieval [CBB19] or graph matching in general [LGD+19]. Here, the parameters are shared over several GNN and the similarity metric is learnt on the fixed-size graph embedding as illustrated in Fig. 4(e).

2.2 Training Set Selection

2.2.1 Pairs

The selection strategy for training examples depends mostly on the application and the kind of knowledge about similarities that one wants to incorporate in the model. For many applications, such as face or signature verification, the similarity between examples depends on their "real-world" origin, *i.e.* faces/signatures from the same person, and the neural network allows to determine the genuineness of a test example w.r.t. a reference by means of a binary classification. Most approaches use pairs of training examples $(\mathbf{x}_1, \mathbf{x}_2)$ and a binary similarity relation which takes different values for similar and dissimilar pairs (*c.f.* illustration in Fig. 3). In principle, the overall loss is defined as the sum of loss terms over all possible pairs of training instances (see Section 2.3 for common loss functions). However, this creates a large imbalance as there are usually much more negative pairs than positive ones if the examples are uniformly distributed over classes. And this imbalance may lead to convergence problems because the learning algorithm tries to focus more on the dissimilarities than the similarities, which are not well defined. Thus it may fail to learn a meaningful similarity metric. A straightforward approach to alleviate this problem is to balance the number of positive and negative pairs (*i.e.* using over-sampling/under-sampling) in each training iteration, or alternatively introduce a weight term in the loss function [ZIG+15]. In some cases, it is even beneficial to learn only with positive examples [ZDI+18]. This ensures that during learning the dissimilar

pairs do not impair the learnt projection that brings positive examples closer in the metric space.

2.2.2 Triplets

Lefebvre *et al.* [LG13] proposed to expand the information about the expected neighbourhood, and suggested a more symmetric representation based on the idea of Weinberger *et al.* [WBS06]: by considering a reference example $\mathbf{x}_R$ for each known relation, it is possible to define triplets $(\mathbf{x}_R, \mathbf{x}_+, \mathbf{x}_-)$, with $\mathbf{x}_+$ forming a genuine pair with the reference $\mathbf{x}_R$, while $\mathbf{x}_-$ is an example from another class (impostor) – sometimes also called the *anchor*, the *positive* and the *negative* examples, respectively. In this way, the learnt embedding may correspond to a more balanced representation of similarities and dissimilarities of the data. As for pair-wise SNNs, in principle, the global loss function is defined over all possible triplets (see Section 2.3 for the definition of different triplet loss functions). In practice, however, there are two many possible combinations. Thus, at each training iteration, a subset of all possible triplets are usually randomly and uniformly sampled. One can also define each training example as reference example $\mathbf{x}_R$ in turn, and draw the positive and negative ones randomly. Or if classes are not balanced, first randomly sample a reference example from each class in turn, and then sample the other two.

However, these sampling strategies may not be very efficient because, after some training iterations, most of the triplets are not contributing much (or not at all) to learning the similarity metric because they have already been integrated in the model, and the loss (and gradient) is thus very small or even zero. Therefore, different heuristics have been proposed to improve convergence and also the semantic expressiveness of the learnt similarity metric.

For example, Hermans *et al.* [HBL17] proposed the *hard-batch* triplet loss which is a modification of the common triplet loss (*c.f.* Eq. 8), where, for each reference example, only the hardest positive examples (*i.e.* the ones that are furthest from the reference) and the hardest negative examples (*i.e.* the closest ones) are selected for training. The hard-batch approach is sometimes referred to as the "top-ranking constraint". Wang *et al.* [WZL17] extended this idea with an efficient method for mining hard examples by learning a subspace and clustering the identities, and they applied this approach for large-scale face recognition with 100 000 identities. Schroff *et al.* [SKP15] argued that learning with the hardest negatives may impair the convergence and the performance of the model and proposed a *semi-hard* batch mining strategy which consists in not using those negatives that have a smaller distance to the reference than the positive example in a mini-batch. The deep metric learning formulation from Yu and Tao [YT19] introduce a scaling scheme in their loss where hard triplets are up-weighted and easier triplets are down-weighted. An additional slack variable introduces a margin that prevents too hard negatives from gaining too much importance. Ge *et al.* [GHDS18]

introduce an hierarchical triplet loss, which consists in dynamically adapting the margin m in Eq. 8 and thus incorporates a hierarchical structure in the embedding that is in turn constructed using the learnt distance metric. This allows to select meaningful hard negatives and to improve the convergence and performance. Another approach called RankTriplet has been proposed by Chen *et al.* [CDS+18] for ranking in a retrieval application. Here hard triplets are selected in subsets (training batches) according to their ranks w.r.t. to a query, *i.e.* only mis-ranked positive and negative examples are used. This allows to define a list-wise measure based on the similarity ranking. Also the effective mining of appropriate positive examples has been studied in the literature. Shi *et al.* [SYZ+16], for instance, propose a *moderate positive mining* strategy that excludes very hard positives examples as their Euclidean distance can differ considerably w.r.t. the geodesic distance on a complex manifold that the learnt metric should correspond to.

2.2.3 Tuples

Several SNN approaches that go beyond triplet architectures have been proposed in the past. For instance, the model introduced by Yih *et al.* [YTPM11] uses a loss based on two pairs of training examples (see Eq. 12), where the first pair is considered more similar than the second one. For some applications, this type of *relative* similarity may be easier to define than the binary relation between similar and dissimilar pairs. Chen *et al.* [CCZH17a] propose a quadruplet loss (*c.f.* Eq. 14) introducing an additional constraint to triplets that also pushes away negatives pairs from positive pairs w.r.t. to different reference images. They experimentally showed that this approach reduces the intra-class variance while increasing the inter-class variance, thus improving its discrimination capacity.

Another possibility proposed by Berlemont *et al.* [BLDG15] is to use tuples $(\mathbf{x}_R, \mathbf{x}_+, \mathbf{x}_{-,1}, \mathbf{x}_{-,2}, \dots, \mathbf{x}_{-,K})$, with a reference $\mathbf{x}_R$, a positive example $\mathbf{x}_+$ and K negative examples $\mathbf{x}_{-,k}$, as illustrated in Fig. 5. Here, the loss function $\mathcal{L}_W$ (Eq. 15) is defined on a modified triplet cosine similarity that tries to bring closer the positive and separate all the negative examples from the reference. This objective essentially operates on $K+1$ pairs of examples. Similarly, Song *et al.* [SXJS16] proposed a deep metric learning approach based on CNNs, where training batches are composed of several positive and negative pairs and the loss function (Eq. 17) brings closer similar pairs and separates each positive pair from its hardest negatives. Later, Yang *et al.* [YCS19] extended this approach defining a loss function on all positive and negative pairs in a tuple (Eq. 18) and a regularisation term preventing the variances of positive and negative distances from becoming too large. They also introduced a dynamically computed weight that focuses the loss on harder positive pairs.

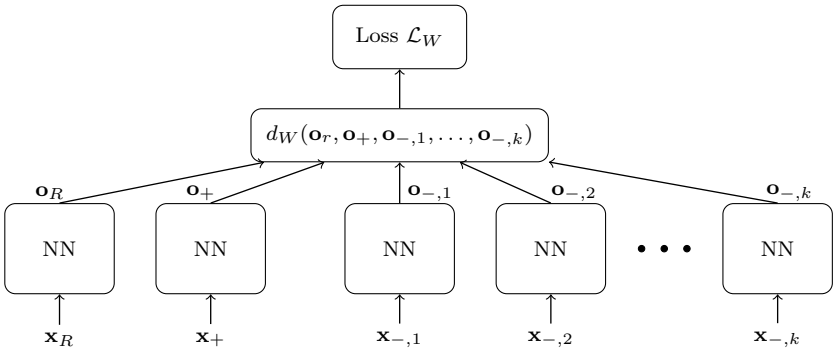

Fig. 5 A SNN architecture trained with tuples composed of a reference example $\mathbf{x}_R$, a positive example $\mathbf{x}_+$, and K negative examples $\mathbf{x}_{-,1}, \ldots, \mathbf{x}_{-,K}$. For $K = 1$, this corresponds to the popular triplet architecture.

Going beyond pair-wise loss formulations, Berlemont *et al.* [BLDG16, BLDG18] extended their previous approach by proposing a loss based on the polar sine function [LW09], a generalisation of the sine function to n dimensions (Eq. 22). The relation between all examples in the tuple is thus taken into account by maximising the hyper-volume spanned by the projected reference and the negative example vectors. By sampling each negative example in a tuple from a different class (different from the one of the reference example) the training becomes more balanced and the learnt similarity metric tends to better reflect the relationship between the classes.

The various SNN architectures and corresponding loss functions described here have been used and evaluated for different applications in the literature. It seems there is no universal strategy for the choice of these hyper-parameters, and their performance largely depends on the given application as well as the type and amount of training data.

2.3 Loss Functions

The loss function $\mathcal{L}_W$ that is to be minimised by the optimisation algorithm defines a global objective in terms of the distances d_W between the neural network output vectors of the training examples $\mathbf{o}_i$ $i \in 1..N$, *i.e.* the projections in the embedding. The loss should ensure that the distances between similar examples is small and between dissimilar examples large. As described above, this measure is usually defined over pairs, triplets or tuples in the loss function, and mostly based on the Euclidean or cosine distance. When minimising the empirical loss over the training examples by iteratively updating the shared weights W of the neural network, one essentially learns a projection function of inputs $\mathbf{o} = f(\mathbf{x})$ into an embedding that forms a similarity

metric space where similar examples lie close (in terms of the defined distance d_W) and dissimilar examples are further apart.

In the following, we will briefly describe some of the most commonly used loss functions. For convenience and unless stated otherwise, we only note the loss on a given subset (pair, triplet, tuple etc.), and the overall loss is usually defined as the sum of these losses over all possible subsets.

Cosine pair-wise

Given a network with weights W and two examples $\mathbf{x}_1$ and $\mathbf{x}_2$ with their labels Y, a target $t(Y)$ is defined for the cosine value between the two respective output vectors $\mathbf{o}_1$ and $\mathbf{o}_2$ as "1" for similar pairs and "-1" (or "0") for dissimilar pairs [BGL+94]:

$$\mathcal{L}_W(\mathbf{x}_1, \mathbf{x}_2, Y) = \left(t(Y) - \cos(\mathbf{o}_1, \mathbf{o}_2)\right)^2 . \tag{1}$$

A similar function is used in the Cosine Similarity Metric Learning (CSML) approach [HL11]:

$$\mathcal{L}_W(\mathbf{x}_1, \mathbf{x}_2, Y) = -t(Y)\cos(\mathbf{o}_1, \mathbf{o}_2) . \tag{2}$$

Triangular

Zheng *et al.* [ZDI+18] use the same targets for a pair-wise function and impose additional constraints on the norms of the output vectors: $\mathbf{o}_1$ and $\mathbf{o}_2$. Integrating a geometrical interpretation using the triangle inequality, the resulting loss function becomes:

$$\mathcal{L}_W(\mathbf{x}_1, \mathbf{x}_2, Y) = \frac{1}{2}\left\|\mathbf{o}_1\right\|^2 + \frac{1}{2}\left\|\mathbf{o}_2\right\|^2 - \left\|\mathbf{o}_1 + t(Y)\,\mathbf{o}_2\right\| + 1 . \tag{3}$$

Norm-based

Several works [CHL05, HCL06, SCWT14, MBBS14] propose to use the norm, *e.g.* ℓ_2-norm, between the output vectors as a similarity measure:

$$d_W(\mathbf{x}_1, \mathbf{x}_2) = \left\|\mathbf{o}_1 - \mathbf{o}_2\right\|_2 . \tag{4}$$

For example, Chopra *et al.* [CHL05] define an objective composed of an "impostor" I (t(Y)=1) and a "genuine" G term (t(Y)=0):

$$\mathcal{L}_W(\mathbf{x}_1, \mathbf{x}_2, Y) = (1 - t(Y))E_W^G(\mathbf{x}_1, \mathbf{x}_2) + t(Y).E_W^I(\mathbf{x}_1, \mathbf{x}_2) \tag{5}$$

$$\text{with } E_W^G(\mathbf{x}_1, \mathbf{x}_2) = \frac{2}{Q}(d_W)^2, E_W^I(\mathbf{x}_1, \mathbf{x}_2) = 2Qe^{(-\frac{2.77}{Q}d_W)} , \tag{6}$$

where Q is the upper bound of d_W.

Many recent works use the so-called *contrastive loss* [HCL06], where

$$E_W^G(\mathbf{x}_1, \mathbf{x}_2) = d_W^2 \quad \text{and} \quad E_W^I(\mathbf{x}_1, \mathbf{x}_2) = \max(m - d_W^2, 0) , \tag{7}$$

with m being a fixed margin parameter.

Triplet

Weinberger *et al.* [WBS06] introduced the triplet loss for a linear metric learning approach using simultaneously targets for genuine and impostor pairs by forming triplets of a reference $\mathbf{x}_R$, a positive $\mathbf{x}_+$ and a negative $\mathbf{x}_-$ example:

$$\mathcal{L}_W(\mathbf{x}_R, \mathbf{x}_+, \mathbf{x}_-) = \max(d_W(\mathbf{o}_R, \mathbf{o}_+)^2 - d_W(\mathbf{o}_R, \mathbf{o}_-)^2 + m, 0) \; . \tag{8}$$

This loss has been used by numerous deep learning approaches in the literature and for various different applications.

Later, Lefebvre *et al.* [LG13] proposed a triplet similarity measure based on the cosine distance. Here, the output of the positive pair $(\mathbf{o}_R, \mathbf{o}_+)$ is trained to be collinear, whereas the output of the negative pair $(\mathbf{o}_R, \mathbf{o}_-)$ is trained to be orthogonal. Thus:

$$\mathcal{L}_W(\mathbf{x}_R, \mathbf{x}_+, \mathbf{x}_-) = (1 - \cos(\mathbf{o}_R, \mathbf{o}_+))^2 + (0 - \cos(\mathbf{o}_R, \mathbf{o}_-))^2 \; . \tag{9}$$

Note that the margin m is a hyper-parameter that is usually empirically determined. It is however not trivial to find its optimal value as it depends mostly on the application, the used dataset and the dimension of the embedding space.

Angular

Wang *et al.* [WZW+17] introduced a loss function based on the angles formed by triplets, called the *angular loss*. Compared to the classical triplet loss, it not only takes into account the reference-positive and reference-negative pairs but also the negative-positive relationship, thus improving its stability during training. It is defined as:

$$.\mathcal{L}_W(\mathbf{x}_R, \mathbf{x}_+, \mathbf{x}_-) = \max(0, ||\mathbf{o}_R - \mathbf{o}_+||^2 - 4\tan^2\alpha||\mathbf{o}_- - \mathbf{o_c}||^2) \; , \tag{10}$$

. where $\mathbf{o}_c = (\mathbf{o}_R + \mathbf{o}_+)/2$, and $\alpha > 0$ is a margin parameter defining an upper bound of the angle between the $\mathbf{o}_-$ and $\mathbf{o}_c$.

Deviance

Yi *et al.* [YLLL14] use the binomial deviance to define their loss function:

$$\mathcal{L}_W(\mathbf{x}_1, \mathbf{x}_2, Y) = \ln\left(\exp^{-2t(Y)\ \cos(\mathbf{o}_1, \mathbf{o}_2)} + 1\right) \; . \tag{11}$$

Quadruplets

Yih *et al.* [YTPM11] consider two pairs of vectors, $(\mathbf{x}_{p1}, \mathbf{x}_{q1})$ and $(\mathbf{x}_{p2}, \mathbf{x}_{q2})$, the first being known to have a higher similarity than the second. The main objective is then to maximise

$$\Delta = \cos(\mathbf{o}_{x_{p1}}, \mathbf{o}_{x_{q1}}) - \cos(\mathbf{o}_{x_{p2}}, \mathbf{o}_{x_{q2}}) \tag{12}$$

in a logistic loss function

$$\mathcal{L}_W(\Delta) = \log(1 + \exp(-\gamma\Delta)) \,, \tag{13}$$

with γ being a scaling factor.

Chen *et al.* [CCZH17b] propose a deep metric learning approach based on quadruplets extending the classical triplet loss (Eq. 8) by introducing an additional constraint that enforces the distance between two negative examples to be larger then between two positive examples but with a different reference (anchor). The overall loss for N training examples is defined as:

$$\mathcal{L}_W = \sum_{i,j,k}^{N} \max(d_W(\mathbf{o}_i, \mathbf{o}_j)^2 - d_W(\mathbf{o}_i, \mathbf{o}_k)^2 + m_1, 0) + \sum_{i,j,k,l]}^{N} \max(d_W(\mathbf{o}_i, \mathbf{o}_j)^2 - d_W(\mathbf{o}_l, \mathbf{o}_k)^2 + m_2, 0) \,, \tag{14}$$

where m_1 and m_2 are two margins, and the input triplets $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ and quadruplets $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l)$ are chosen under label constraints: $y_i = y_j$, $y_l \neq y_k$, $y_i \neq y_l$, $y_i \neq y_k$.

Tuples: pair-wise

The approach from Berlemont *et al.* [BLDG15] is based on tuples $\mathbf{T}$ composed of a reference example $\mathbf{x}_R$, a positive example $\mathbf{x}_+$ and K negative examples $\mathbf{x}_{-,k}\,(k = 1..K)$. Similar to the triplet loss (*c.f.* Eq. 9), with a tuple $\mathbf{T}$, a positive pair $(\mathbf{x}_R, \mathbf{x}_+)$ and K negative pairs are formed, with respective target values "1" and "0" for the cosine distance. Thus, given the respective outputs of the neural network $\mathbf{o}_R, \mathbf{o}_+, \mathbf{o}_{-,1}, \ldots, \mathbf{o}_{-,K}$, the loss for a tuple $\mathbf{T}$ is defined as follows:

$$\mathcal{L}_W(\mathbf{T}) = (1 - \mathbf{o}_R \cdot \mathbf{o}_+)^2 + \sum_{k=1}^{K} (0 - \mathbf{o}_R \cdot \mathbf{o}_{-,k})^2 + \sum_{\mathbf{x}_p \in \mathbf{T}} (1 - \|\mathbf{o}_p\|)^2 \,, \tag{15}$$

where the cosine distances for the positive pair and the K negative pairs have been replaced by the scalar product as $\cos(\mathbf{o}_1, \mathbf{o}_2) = \frac{\mathbf{o}_1 \cdot \mathbf{o}_2}{\|\mathbf{o}_1\| . \|\mathbf{o}_2\|}$, and the norms are enforced to be "1" in the term of the last sum.

A very similar approach has been proposed by Son [Soh16] based on the classical triplet formulation of Weinberger *et al.* (Eq. 8) using the so-called *N-pair loss*:

$$\mathcal{L}_W(\mathbf{T}) = \log\left(1 + \sum_{i=1}^{K} \exp\left(\mathbf{o}_R^\top \mathbf{o}_{-,k} - \mathbf{o}_R^\top \mathbf{o}_+\right)\right) \,. \tag{16}$$

Song *et al.* [SXJS16] proposed another pair-wise formulation based on several positive and several negative pairs, $\mathcal{P}$ and $\mathcal{N}$, respectively. For each positive pair $(\mathbf{x}_i, \mathbf{x}_j)$, only the closest negative examples (for either) have an influence during the training. The initial loss function involves a discontinuous *max* function that is replaced by a smooth upper bound defined by the "log-exp-sum" of the two terms for the positive pair, which leads to the final loss:

$$\mathcal{L}_W = \frac{1}{2|\mathcal{P}|} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{P}} \max(0, \mathcal{L}_W^{(i,j)})^2$$

$$\mathcal{L}_W^{(i,j)} = \log\left(\sum_{(\mathbf{x}_i,\mathbf{x}_k)\in\mathcal{N}} \exp(m - d(\mathbf{x}_i,\mathbf{x}_k)) + \sum_{(\mathbf{x}_j,\mathbf{x}_l)\in\mathcal{N}} \exp(m - d(\mathbf{x}_j,\mathbf{x}_l)) \right) + d(\mathbf{x}_i,\mathbf{x}_j) \ , \quad (17)$$

where $d(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{x}_i - \mathbf{x}||_2$ is the Euclidean distance, and m is a margin.

Yang *et al.* [**?**] proposed a similar "structural" loss function based on all positive and negative pairs, $\mathcal{P}$ and $\mathcal{N}$, in a tuple (mini-batch) $\mathbf{T}$:

$$\mathcal{L}_W(\mathbf{T}) = \frac{1}{\mathcal{B}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{P}} \beta_{ij} \log\left(1 + \sum_{(\mathbf{x}_i,\mathbf{x}_k)\in\mathcal{N}} \exp\left(d(\mathbf{x}_i,\mathbf{x}_j)^2 - d(\mathbf{x}_i,\mathbf{x}_k)^2 + m\right)/\xi \right) + \frac{\lambda}{2}(|\sigma_p^2 - m_p]_+ + [\sigma_n^2 - m_n]_+) \ , \quad (18)$$

where $\xi < 1$ is a constant making the triplet constraint "softer", m, m_p and m_n are margin parameters, and $[x]_+ = \max(0, x)$ is a hinge loss minimising the second-order statistics σ_p and σ_n of positive and negative pair distances. This second term serves as a regularisation (controlled by factor λ) preventing the variances of positive and negative distances from becoming too large. The factor β_{ij} (with $\mathcal{B} = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{P}} \beta_{ij}$) weights each positive pair $(\mathbf{x}_i, \mathbf{x}_j)$ according to its "hardness", *i.e.* the distance in the Euclidean space:

$$\beta_{ij} = \exp(d(\mathbf{x}_i,\mathbf{x}_j)^2 - \tau_c) \ , \quad (19)$$

with τ_c being a class-dependent threshold, defined as twice the mean minus the minimum of positive distances in $\mathcal{P}$.

Tuples: polar sine

Later, Berlemont *et al.* [BLDG16, BLDG18] proposed a loss function based on tuples that simultaneously takes into account the relationship between *all* examples in the tuple, *i.e.* also between the negatives and not only pair-wise w.r.t. to a reference.

Inspired by the 2D sine function, Lerman *et al.* [LW09] define the polar sine for a set $V_m = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ of m-dimensional linearly independent vectors

($m > n$) as a normalised hyper-volume. Given $\mathbf{A} = \begin{bmatrix}\mathbf{v}_1 \ \mathbf{v}_2 \cdots \mathbf{v}_m\end{bmatrix}$ and its transpose $\mathbf{A}^\top$:

$$PolarSine(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \frac{\sqrt{\det\left(\mathbf{A}^\top \mathbf{A}\right)}}{\prod_{i=1}^{n} \|\mathbf{v}_i\|}. \tag{20}$$

In the special case where $m = n$, the matrix product in the determinant is replaced by the square matrix $\mathbf{A}$.

Given the matrix $\mathbf{S}$ such that $\forall (i,j) \in [1,..,n]^2, \mathbf{S}_{i,j} = \cos(\mathbf{v}_i, \mathbf{v}_j)$, this measure can be rewritten as $PolarSine(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \sqrt{\det\left(\mathbf{S}\right)}$. For numerical stability reasons during the derivation process, and to make this value independent from the number of classes, the authors define a polar sine metric as follows:

$$psine(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \sqrt[n]{\det\left(\mathbf{S}\right)}. \tag{21}$$

This metric only depends on the angles between every vector of the set. It reaches its maximum value when all the vectors are orthogonal, and thus can be used as a measure for dissimilarity. The loss function for a tuple $\mathbf{T}$ is defined as:

$$\begin{aligned} \mathcal{L}_W(\mathbf{T}) &= \mathcal{L}_W^p(\mathbf{T}) + \mathcal{L}_W^n(\mathbf{T}) \ , \\ \mathcal{L}_W^p(\mathbf{T}) &= (1 - \cos(\mathbf{o}_R, \mathbf{o}_+))^2, \\ \mathcal{L}_W^n(\mathbf{T}) &= (1 - psine(\mathbf{o}_R, \mathbf{o}_{-,1}, \ldots, \mathbf{o}_{-,K}))^2 \ . \end{aligned} \tag{22}$$

Optimising the polar sine corresponds to assigning a target of 0 to the cosine value of every pair of outputs from different vectors drawn in $\mathbf{T} \setminus \{\mathbf{x}_R\}$, *i.e.* a target for every pair of dissimilar examples.

SoftTriple

The approach proposed by Qian *et al.* [QSS+19] uses a loss formulation based on the SoftMax function. They showed that the SoftMax loss, commonly used in supervised learning, is essentially equivalent to a smoothed triplet loss, where each class has a single "centre", and they extended this to multiple centres allowing to cope with large intra-class variations and underlying multi-modal distributions. The similarity metric is learnt in a supervised way with C classes and an additional fully-connected layer l with weight vectors $[\mathbf{w}_1, \ldots, \mathbf{w}_C] \in \mathbb{R}^{d \times C}$ before the final SoftMax layer. The algorithm minimises the following loss defined on a single input $\mathbf{x}_i$ to the layer l and its label y_i over the entire training set:

$$\mathcal{L}_W = -\log \frac{\exp(\lambda(S_{i,y_i} - \delta))}{\exp(\lambda(S_{i,y_i} - \delta)) + \sum_{j \neq y_i} \exp(\lambda S_{i,j})} \tag{23}$$

where all weights $\mathbf{w}_i$ and $\mathbf{x}_i$ have unit length, λ is a regularisation constant, δ is a margin, and

$$S_{i,c} = \sum_k \frac{\exp(\frac{1}{\gamma}\mathbf{x}_i^\top w_c^k)}{\sum_k \exp(\frac{1}{\gamma}\mathbf{x}_i^\top w_c^k)} \mathbf{x}_i^\top w_c^k \tag{24}$$

is a relaxed (hence *soft*) version of

$$S'_{i,c} = \max_k \mathbf{x}_i^\top w_c^k \ , \tag{25}$$

a similarity between the input $\mathbf{x}_i$ and the (closest) k-th centre (*i.e.* weight vector) of class c. The final embedding representing the similarity metric is thus produced at the output of layer l.

Sphere Loss

Similar to the SoftTriple loss, Fan *et al.* [FJLF19] proposed a similarity metric learning approach based on the SoftMax function, where, again, the embeddings $\mathbf{o}_i$ and weights $\mathbf{w}_i$ $i = 1..C$ of the last layer are set to unit norm and thus lie on a hyper-sphere. Here, the similarity is expressed in terms of the angle θ_j between the weight ("centre") $\mathbf{w}_j$ and an embedding o. And the loss for one example $\mathbf{x}_i$ with label $\mathbf{y}_i$ is defined as:

$$\mathcal{L}_W = -\log \frac{e^{s\ \cos\theta_{y_i}}}{\sum_{j=1}^{C} e^{s\ \cos\theta_j}} \ , \tag{26}$$

where s is a scaling factor. Thus, after training, the embedding is formed by the hyper-sphere manifold described by the last layer outputs, and similarity is measured in terms of the angular distance.

Probability-driven

Nair *et al.* [NH10] add a final unit to their neural network architecture whose activation function computes the probability P of two examples $\mathbf{x}_1$, $\mathbf{x}_2$ being from the same class:

$$P = \frac{1}{1 + \exp(-(w.\cos(\mathbf{o}_1, \mathbf{o}_2) + b))} \ , \tag{27}$$

with w and b being scalar parameters.

Statistical

Chen *et al.* [CS11] compute the first and second-order statistics, $\mu^{(i)}$ and $\Sigma^{(i)}$, over sliding windows on the SNN outputs of a speech sample i, and define the loss function as:

$$\mathcal{L}_W(\mathbf{x}_1, \mathbf{x}_2, Y) = (1 - t(Y))(D_m + D_S) + t(Y).(\exp(\frac{-D_m}{\lambda_m}) + \exp(\frac{-D_S}{\lambda_S})) \ , \tag{28}$$

where

$$D_m = \left\| \mu^{(i)} - \mu^{(j)} \right\|_2^2 \quad , \quad D_S = \left\| \Sigma^{(i)} - \Sigma^{(j)} \right\|_F^2 \tag{29}$$

are incompatibility measures of these statistics between two samples i and j, λ_m and λ_s are tolerance bounds on these measures, and $\|.\|_F$ is the Frobenius norm.

2.4 Training Algorithms and Schemes

In principle, any optimisation algorithm for neural networks can be used to train a SNN. In most similarity learning approaches from the literature, the standard Stochastic Gradient Descent (SGD) with mini-batches has been used. The batch size is sometimes an important hyper-parameter because it may have an influence on the overall convergence. Furthermore, as mentioned in Section 2.2, the selection of examples used for one training iteration might be pre-determined by the SNN architecture or loss function. For instance, for an SNN with a pair-wise loss function, *e.g.* contrastive divergence, different pairs can be formed with the examples in the batch, and it is often recommended to balance the possible number of positive and negative pairs. For triplets, it may be appropriate to have at least one positive/similar pair for each example. For tuples, there may need to be at least one example for each class. In any case, at each training iteration, the loss function (*c.f.* Section 2.3) is generally evaluated on the set of examples in the batch. And the gradients are computed for each input example and summed. Then the (shared) weights are updated according to the standard error backpropagation algorithm.

To circumvent the problem of exhaustive triplet sampling, Movshovitz-Attias *et al.* [MATL+17] introduced the notion of *proxies*, where a representative point (proxy) from a much smaller subset is assigned to each positive and negative training example in the triplets as an approximation. The proxies are updated during training, and the algorithm thus requires much less memory and further improves the convergence.

Another original similarity metric learning algorithm proposed by Duan *et al.* [DZL+18] uses a Generative Adversarial Network (GAN) model to generate synthetic hard negative examples for efficient learning of the metric space. The generator loss encourages the creation of realistic negative examples that are as close as possible to the positive ones and that violate the triplet constraint. The algorithm jointly minimises this loss with the metric loss, and the authors experimented with various loss functions: contrastive, triplet, N-pair tuples *etc.*, and obtained improved image retrieval results using the N-pair loss (*c.f.* Eq. 16).

For large embeddings and complex, deep SNN models, the convergence can be difficult when learning only with similarity and dissimilarity constraints. For this reason, some approaches suggest to either pre-train the model in a supervised way using the labels of the training dataset and a temporary final fully-connected SoftMax layer [CS11] or to simultaneously train

identities and the metric space using the combination of two (or more) loss functions [MdRM17, CCZH17b, YWLY18, MMLJ20].

Finally, when the metric that is to be learnt is applied to a ranking problem, like image retrieval for example, additional constraints on this ranking can be imposed. For example, the approach proposed by Chen *et al.* [CDS+18] integrates the classical mean average precision and rank-1 measures as a weighting term in the loss function and specifically selects mis-ranked triplets in the training batches in order to correct ranking errors. Another approach for deep metric learning to rank, called *FastAP*, has been presented by Cakir *et al.* [CHX+19]. Here, instead of sampling pairs or triplets of training examples, the training is directly optimising the mean average precision within mini-batches by computing the distance histograms between each query and sets of gallery examples in the batch. Using a differentiable soft-binning technique, gradient descent can directly be performed on these histograms of positive and negative retrieval results.

3 Conclusion

In this chapter, we gave an overview on the principal models and training algorithms for similarity metric learning with neural networks. Numerous variants of loss functions, neural architectures and training sample selection have been proposed in the literature, and these choices mostly depend on the given training data, the task to solve and the application domain. The general trend seems to go away from simple pair-wise training and triplets to samples involving tuples of several similar and dissimilar examples allowing to model richer structural relations, but these strategies are nowadays mostly based on application-dependant heuristics involving empirical parameters. More research work needs to be done to study the generalisation capacity and the general applicability (beyond person re-identification, for example) for more complex state-of-the-art deep metric learning models. In this regard, the interplay between classification, feature extraction and the learnt metric needs to be better understood. Finally, the notion of similarity (and dissimilarity) is to be defined more formally and extended, possibly by introducing other (prior) knowledge, in order design new models and algorithms that learn a rich, abstract and semantic representation of training data.

References

[ASS20] Mohammad Adiban, Hossein Sameti, and Saeedreza Shehnepoor. Replay spoofing countermeasure using autoencoder and siamese networks on ASVspoof 2019 challenge. *Computer Speech & Language*, 64, 2020.

[BGL+94] Jane Bromley, Isabelle Guyon, Yann Lecun, Eduard Säckinger, and Roopak Shah. Signature Verification using a "Siamese" Time Delay Neural Network. In *Proceedings of NIPS*, 1994.

[BHS13] A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *Computing Research Repository*, abs/1306.6709, 2013.

[BLDG15] Samuel Berlemont, Gregoire Lefebvre, Stefan Duffner, and Christophe Garcia. Siamese neural network based similarity metric for inertial gesture classification and rejection. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition (FG)*, Ljubljana, Slovenia, May 2015.

[BLDG16] Samuel Berlemont, Grégoire Lefebvre, Stefan Duffner, and Christophe Garcia. Polar sine based siamese neural network for gesture recognition. In *Proceedings of the International Conference on International Conference on Artificial Neural Networks (ICANN)*, Barcelona, Spain, 2016.

[BLDG18] Samuel Berlemont, Grégoire Lefebvre, Stefan Duffner, and Christophe Garcia. Class-balanced siamese neural networks. *Neurocomputing*, 273:47–56, 2018.

[BWCB11] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, 2011.

[CBB19] Ushasi Chaudhuri, Biplab Banerjee, and Avik Bhattacharya. Siamese graph convolutional network for content based remote sensing image retrieval. *Computer Vision and Image Understanding*, 184:22 – 30, 2019.

[CCZH17a] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[CCZH17b] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. A multi-task deep network for person re-identification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[CDS+18] Yiqiang Chen, Stefan Duffner, Andrei Stoian, Jean-Yves Dufour, and Atilla Baskurt. Similarity learning with listwise ranking for person re-identification. In *Proceedings of the International Conference on Image Processing (ICIP)*, 2018.

[CHL05] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 539–546. IEEE, 2005.

[CHX+19] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[CLDG19] Paul Compagnon, Gregoire Lefebvre, Stefan Duffner, and Christophe Garcia. Routine modeling with time series metric learning. In *Proceedings of the International Conference on International Conference on Artificial Neural Networks (ICANN)*, September 2019.

[CS11] Ke Chen and Ahmad Salman. Extracting Speaker-Specific Information with a Regularized Siamese Deep Network. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 298–306, 2011.

[CSSB10] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135, 2010.

[DZL+18] Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep adversarial metric learning. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[FJLF19] Xing Fan, Wei Jiang, Hao Luo, and Mengjuan Fei. SphereReID: Deep hypersphere manifold embedding for person re-identification. *Journal of Visual Communication and Image Representation*, 60:51–58, 2019.

[GHDS18] Weifeng Ge, Weilin Huang, Dengke Dong, and Matthew R. Scott. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[HBL17] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In Defense of the Triplet Loss for Person Re-Identification. *arXiv preprint arXiv:1703.07737*, 2017.

[HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742, 2006.

[HL11] N. V. Hieu and B. Li. Cosine similarity metric learning for face verification. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 709–720. Springer, 2011.

[HLT14] J. Hu, J. Lu, and Y.-P. Tan. Discriminative deep metric learning for face verification in the wild. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1875–1882, 2014.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[KBR16] Theofanis Karaletsos, Serge Belongie, and Gunnar Rätsch. Bayesian representation learning with oracle constraints. In *International Conference on Learning Representations (ICLR)*, 2016.

[KTS+12] D. Kedem, S. Tyree, F. Sha, G. R. Lanckriet, and K. Q. Weinberger. Nonlinear metric learning. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 2573–2581, 2012.

[LBOM12] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[LG13] Grégoire Lefebvre and Christophe Garcia. Learning a bag of features based nonlinear metric for facial similarity. In *Proceedings of the International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, pages 238–243, 2013.

[LGD+19] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

[LW09] Gilad Lerman and J. Tyler Whitehouse. On D-dimensional D-semimetrics and simplex-type inequalities for high-dimensional sine functions. *Journal of Approximation Theory*, 156(1):52–81, January 2009.

[MATL+17] Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[MBBS14] Jonathan Masci, Michael M. Bronstein, Alexander M. Bronstein, and Jurgen Schmidhuber. Multimodal Similarity-Preserving Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):824–830, April 2014.

[MdRM17] Niall McLaughlin, Jesus Martinez del Rincon, and Paul C Miller. Person reidentification using deep convnets with multitask learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(3):525–539, 2017.

[MLK16] Panagiotis Moutafis, Mengjun Leng, and Ioannis A Kakadiaris. An overview and empirical comparison of distance metric learning methods. *IEEE Transactions on Cybernetics*, 2016.

[MMLJ20] Weiqing Min, Shuhuan Mei, Zhuo Li, and Shuqiang Jiang. A two-stage triplet network training framework for image retrieval. *IEEE Transactions on Multimedia*, 2020.

[MT16] J. Mueller and A. Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[NH10] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

[NVR16] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning Text Similarity with Siamese Recurrent Networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157, August 2016.

[QGCL08] A. M. Qamar, E. Gaussier, J. P. Chevallet, and J. H. Lim. Similarity learning for nearest neighbor classification. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 983–988. IEEE, 2008.

[QSS+19] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. SoftTriple loss: Deep metric learning without triplet sampling. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[SCWT14] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1988–1996, 2014.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[SL20] Weijie Sheng and Xinde Li. Siamese denoising autoencoders for joints trajectories reconstruction and robust gait recognition. *Neurocomputing*, 395:86 – 94, 2020.

[Soh16] Kihyuk Sohn. Improved deep metric learning with multi-class N-pair loss objective. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2016.

[SXJS16] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[SYZ+16] Hailin Shi, Yang Yang, Xiangyu Zhu, Shengcai Liao, Zhen Lei, Weishi Zheng, and Stan Z. Li. Embedding deep metric for person re-identification: A study against large variations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[WBS06] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, volume 18, page 1473, 2006.

[WZL17] Chong Wang, Xue Zhang, and Xipeng Lan. How to train triplet networks with 100k identities? In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[WZW+17] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep metric learning with angular loss. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[XNJR03] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 521–528. MIT; 1998, 2003.

[YCS19] Yao Yang, Haoran Chen, and Junming Shao. Triplet enhanced autoencoder: Model-free discriminative network embedding. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[YLLL14] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Deep metric learning for person re-identification. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, pages 34–39, 2014.

[YT19] Baosheng Yu and Dacheng Tao. Deep metric learning with tuplet margin loss. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[YTPM11] Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 247–256. Association for Computational Linguistics, 2011.

[YWLY18] Mang Ye, Zheng Wang, Xiangyuan Lan, and Pong C. Yuen. Visible thermal person re-identification via dual-constrained top-ranking. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[YYGT16] Jun Yu, Xiaokang Yang, Fei Gao, and Dacheng Tao. Deep multimodal distance metric learning using click constraints for image ranking. *IEEE Transactions on Cybernetics*, 2016.

[ZDI+18] Lilei Zheng, Stefan Duffner, Khalid Idrissi, Christophe Garcia, and Atilla Baskurt. Pairwise identity verification via linear concentrative metric learning. *IEEE Transactions on Cybernetics*, 48(1):324–335, 2018.

[ZIG+15] Lilei Zheng, Khalid Idrissi, Christophe Garcia, Stefan Duffner, and Atilla Baskurt. Logistic similarity metric learning for face verification. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2015.