A Robust Distance Metric for Deep Metric Learning

Authors (Matriculation Numbers):
Ziyang Qiu (3565830) and Thomas Rekers (3489800)

Report for Advanced Machine Learning by Prof. Ulrich Köthe

20th September 2019
Heidelberg University

Contents

1 Executive Summary

Deep metric learning deals with the learning of discriminative features to process classification tasks and image clustering. It is also called similarity learning and gained increasing attention in recent years. The basic idea of deep metric learning is that the image input data is mapped by a neural network into a multidimensional feature space, where similar samples are mapped close to each other and dissimilar samples are mapped more distant to each other. This report is based on the paper "Signal-to-Noise Ratio: A Robust Distance Metric for Deep Metric Learning"[1]. At first we will summarize the paper by explaining the two different approaches in metric learning, the structure learning and the distance learning, and introducing the robust SNR distance metric, which is based on Signal-to-Noise Ratio (SNR) for measuring the similarity of images. We give explanations regarding the modifications in the loss functions for training models with the SNR metric and discuss the results from the experiments based on AlexNet in the paper. Afterwards we introduce our own experiments based on AlexNet and ResNet50 and present the differences between these two network models. As explained in the paper the SNR metric can be considered as a relative Euclidian metric under special circumstances. In this case relative Euclidian metric means that the ratio between the Euclidian distance of two samples and the Euclidian distance of one sample to the origin of the feature space is returned. Therefore we decided to design an analogous relative metric not based on the Euclidian metric but on the Mahalanobis metric and call it relative Mahalanobis metric. We summarize the properties of the Mahalanobis metric as described in the paper "Distance Metric Learning for Large Margin Nearest Neighbor Classification"[2] and we present the resuts of our experiments including the self-designed relative Mahalanobis metric. Finally we discuss our results by comparing the scores of all metrics used in the experiments.

2 Introduction

The fundamental idea of metric learning is to map the input data, in most cases a set of images, into a multidimensional feature space. Similar samples are mapped close to each other in the feature space and dissimilar samples are mapped farther apart. This approach can be used both for classification and clustering tasks. While in conventional metric learning the images are mapped linearly into the feature space, in deep metric learning we use deep neural network models like AlexNet or ResNet to get a nonlinear embedding of the input data.

Because of its great successes in last years deep metric learning is widely applied in face recognition, where the model has to be trained with a very large number of different classes. The contrary approach to deep metric learning is to design a network architecture where as many neurons are in the output layer as different classes are in the dataset. In this case, the model identifies each neuron in the output layer with one class of the dataset and the classification is done by selecting the neuron with the largest

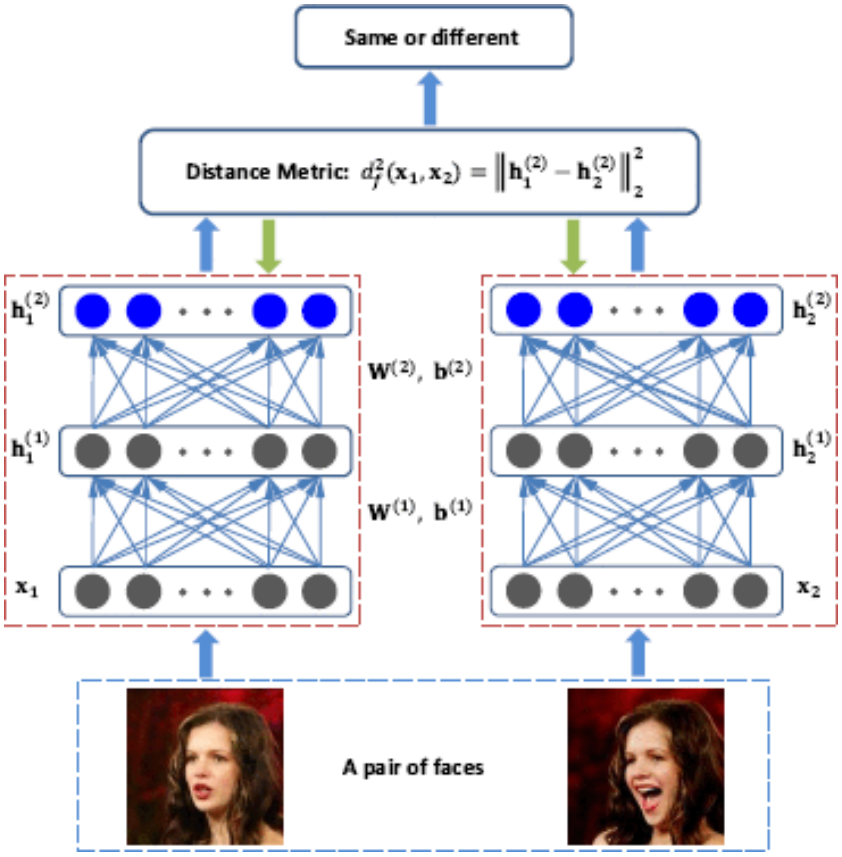


Figure 1: The flowchart shows the frame work of DML. A given pair of face images x_1 and x_2 are mapped into the same feature space as h_1 and h_2 through deep neural networks, where their similarity is calculated.

output. While this approach becomes problematically in case of a large number of different classes and a small number of samples per class, deep metric learning concepts still show good results in such cases.

In case of many different classes the number of classes is usually greater than the number of neurons in the output layer in deep metric learning. Therefore we cannot predict the class by identifying each class with one neuron. Instead the classification follows from the distances in the feature space. For example, the k-Nearest-Neighbor algorithm can be chosen to assign a sample to its class.

In deep metric learning we can distinguish between two different learning approaches. The first and most common approach is structure learning. Structure learning tries to construct more effective structures for the loss functions. For example, many neural network architectures include sampler functions which deliver a customized batch of samples from the feature space to the loss function. Varying the batch size or increasing the number of negative labeled samples in a batch can be strategies in structure learning. The loss function is designed according to the shape of the batch for the backward

training. We will introduce such loss functions and the shape of their batches in the preliminaries. It is to notice that all loss functions use metrics to get a semantic distance between an anchor sample and a comparison sample in the feature space. But structure learning approaches just try to optimize the sampling for the loss function or the summation and weights inside of the loss function. The metric is not changed in structure learning but is changed in distance learning, what is the second approach of metric learning.

The most common learning processes use the Euclidian metric to calculate the distances of the samples in the feature space. In contrast, distance learning tries to get longer distances between similar and shorter distances between dissimilar samples in the feature space by choosing a suitable metric inside of the loss function.[1]

3 Preliminaries

3.1 Loss Functions

As mentioned before optimizing a neural network by tuning the objective function of the training process is part of structure learning. The loss functions introduced in this chapter are based on the Euclidian metric. We will see later that the loss functions for other metrics look similar, but in some cases we have to add a regularization term.

3.1.1 Contrastive Loss

The most common structure learning approach is constrastive embedding with its objective function contrastive loss. The idea is to introduce a margin so that dissimilar samples are pushed apart by this given margin. Similar samples are pulled as close as possible to each other. The objective function for contrastive loss is

$$L = \frac{1}{2N} \sum_{(i,j)\in\Gamma}^{N} \left(y d_{i,j}^2 + (1-y) \max\left(margin - d_{ij}\right)^2 \right) , \tag{1}$$

where d_{ij} is the Euclidean distance between the sample pairs in the feature space. We have $y = 1$ if two samples are in the same class and $y = 0$ otherwise. The $margin$ is the threshold to evaluate whether a pair of samples is in the same class or not. Therefore d_{ij} should be greater than the margin in case of a pair of unsimilar samples and less than the margin in case of two samples of the same class. Γ is the set of the samples and N is the size of the set.[3]

3.1.2 Triplet Loss

Another common objective function is triplet loss. In triplet loss the loss function operates on a set of triplets, where each triplet consists of an anchor sample, a positive sample and a negative sample. Positive means that the considered sample has the same

class as the anchor. Otherwise the sample is negative. The objective function affects that the distance between the anchor and the positive sample is smaller than the distance between the anchor and a negative sample with a margin. The formula for triplet loss is

$$L = \frac{1}{|\Gamma|} \sum_{(i,j,k)\in\Gamma} \max(d_{ij}^2 + m - d_{ik}^2, 0)\,, \tag{2}$$

where i is the index of the anchor, j the index of the positive sample and k the index of the negative sample. d is the Euclidian metric again and Γ the set of the samples. [**?**]

3.1.3 N-Pair Loss

N-pair loss is a generalization of triplet loss by exploiting the possibility to interact with negative samples of all classes. This means that we deal with one positive sample from the same class as the anchor and with $N-1$ negative samples, where every negative sample is from a different class of the $N-1$ other classes. The formula for N-pair loss is

$$L = \frac{1}{N} \sum_{i=1}^{N} \log\left(1 + \sum_{j\neq i} \exp\left(f_i' f_j^+ - f_i' f_i^+\right)\right)\,, \tag{3}$$

where $f_i = f(x_i)$ and $\{(x_i, x_i^+)\}_{i=1}^N$ are N pairs of input samples from N different classes with x_i as anchors and x_i^+ as their positive samples. It applies for their labels y_i that $y_i \neq y_j \,\forall\, i \neq j$ and therefore $\{x_j^+, j \neq i\}$ are the negative input samples to x_i.

3.1.4 Lifted Structured Loss

In lifted structured loss all negative samples are incorporated through training. The positive samples are pulled as close as possible to the anchor and the negative samples are pushed away farther than a given margin. The formula for structured lifted loss is

$$L = \frac{1}{2|P|} \sum_{(i,j)\in P} \max\left(L_{ij}, 0\right)^2 \tag{4}$$

$$\text{with} \quad L_{ij} = \log\left(\sum_{(i,k)\in N} \exp\left(\alpha - d_{ik}\right) + \sum_{(j,l)\in N} \exp\left(\alpha - d_{jl}\right)\right) + d_{ij}\,, \tag{5}$$

where α is the margin parameter. N denotes the set of negative pairs and P denotes the set of positive pairs.

3.2 Distance Metrics

Measuring similarities between pairs of samples have been a research topc for many years. The most well-known distance metric undoubtedly is Euclidian distance, fwhich has

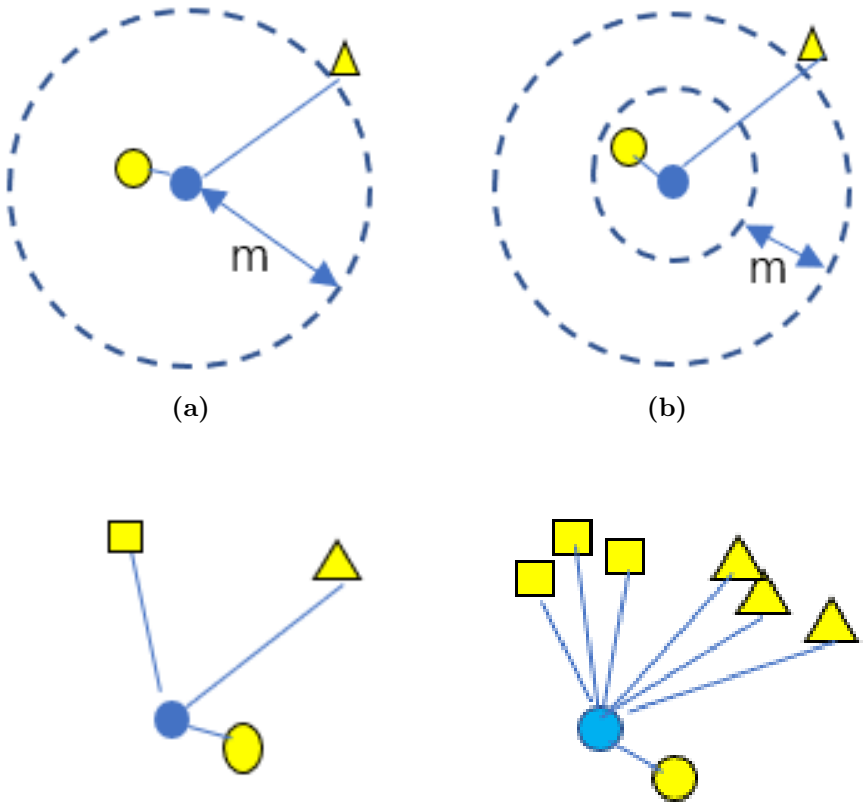


Figure 2: Illustration of (a) contrastive loss, (b) triplet loss, (c) N-pair loss and (d) lifted structured loss. The blue circle is anchor, different shapes indicate different classes.

been widely used in learning discriminative embeddings. But Euclidean distance metric only measures the distance between paired samples in n-dimensional space, and fail to preserve ghe correlation and improve the robustness of the pairs. Some other distance metrics have been proposed to avoid the weakness of Euclidean distance. One of those is Mahalanobis Distance, which is based on correlations between varaibles by which different patterns can be identified and analyzed. Recently, many new improvements in metric learning have been proposed. To improve the generalization of the learned features, Chen et al. introduced the energy confusion metric to confuse the network metric. Taking the angular relationship between samples into account, Jian Wang et al. introduced an Angular Loss improving the robustness of objective against feature variance. A new framework is also being explored. For example, Xinshao Wang et al. proposed ranked list loss that could better make use of datasets by constructing lists consisting of weighted negative examples and positive examples mined from the training

set. Moreover, instead of using distance metrics, Flood Sung et al. managed to teach the Relation Network to larn the metric by itself.

4 Approach of the Paper

4.1 Notations and Technical Terms

In deep metric learning a neural network can be considered as function f called learning function, that maps input data like image samples into a multidimensional feature space. We write $h = f(\theta, x)$ where x is an image sample, h is the embedded sample in the feature space and *theta* represents the parameters and weights of the neural network. Given two features h_i and h_j in the feature space, we consider h_i as anchor feature and h_j as compared feature. Then we call h_i signal and h_j noisy signal and define the noise n_{ij} in h_i and h_j as $n_{ij} = h_j - h_i$. The signal-to-noise ratio is defined as the ratio of signal vvariance to noise variance. Therefore we can define the SNR between h_i and h_j as

$$SNR_{i,j} = \frac{\mathrm{Var}(h_i)}{\mathrm{Var}(h_j - h_i)} = \frac{\mathrm{Var}(h_i)}{\mathrm{Var}(n_{ij})} \tag{6}$$

$$\text{with} \quad \mathrm{Var}(a) = \frac{\sum_{i=1}^{n}(a_i - \mu)^2}{n}, \tag{7}$$

where μ denotes the mean of sample a.

4.2 SNR-Based Metric

From the perspective of information theory, a greater signal variance represents more useful information. In contrast the variance of noise can be seen as a measure of useless information. As a result, the greater the SNR is, the greater is the ratio of useful information in ratio to useless information. The most important constraint to a distance metric in deep metric learning is that similar samples shall have a short distance, while dissimilar samples shall have a longer distance. Though, the SNR, as basing on the variance on the samples and their differences, assumes an independent Gaussian distribution in every component of the samples, what might be a wrong assumption in some cases. As we found in the SNR considerations, greater SNR indicates similarity and smaller SNR indicates dissimilarity. Therefore we define an SNR based distance measure as the reciprocal of the SNR and call it SNR metric d_S:

$$d_S(h_i, h_j) = \frac{1}{SNR_{i,j}} = \frac{\mathrm{Var}(n_{ij})}{\mathrm{Var}(h_i)}. \tag{8}$$

In figure 3 we see a comparison between anchor signals and compared signals, that have different SNR distances to the anchor signals. The 32-dimensional anchor features are in a Gaussian distribution with mean zero and variance one. The compared signals are synthesized by adding Gaussian noises with mean zero and different variances σ^2 to the

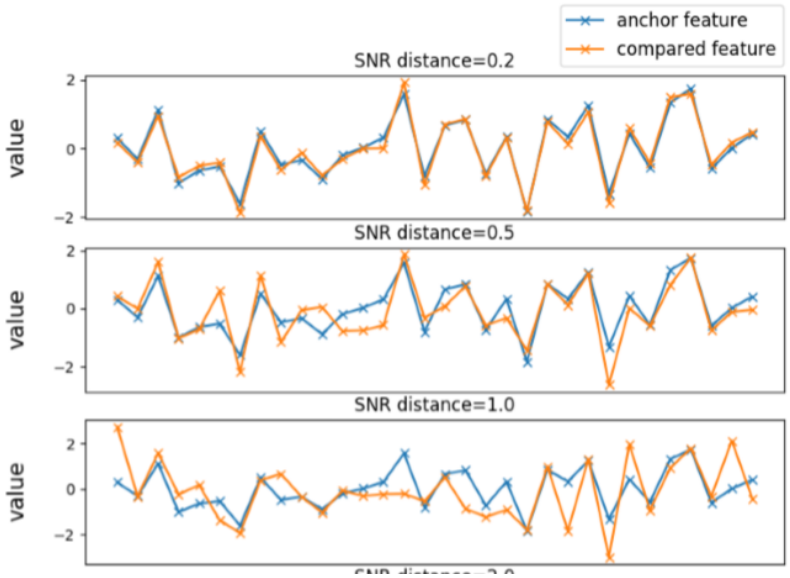


Figure 3: The curves show the comparison of anchor features and the compared features under different SNR distances.

anchor features, where $\sigma^2 = \{0.2, 0.5, 1.0\}$. As we see in the figure, the longer the SNR distances are, the greater the differences between the anchor signals and the compared signals. We notice again that the SNR metric may not be suitable if the anchor samples and compared samples are not in Gaussian distribution or the components of the samples are not independent to each other.

4.3 Superiority Analysis

Now we want to compare the SNR distance to the Euclidean distance. Given two samples in an n-dimensional feature space. Then their Euclidean distance is defined as

$$d_E(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}\,. \tag{9}$$

Assuming a zero-mean-distribution in every component of the samples, two samples h_i and h_j in an n-dimensional feature space have the SNR distance

$$d_S(h_i, h_j) = \frac{\mathrm{Var}(n_{ij})}{\mathrm{Var}(h_i)} = \frac{\sum_{m=1}^{n} (h_{jm} - h_{im})^2}{\sum_{m=1}^{M} h_{im}^2} = \frac{d_E(h_i, h_j)^2}{d_E(h_i)^2}\,, \tag{10}$$

where $d_E(h_i)$ denotes the Euclidean distance from h_i to the origin zero of the feature space. We will continue with the assumption of a zero-mean-distribution. This assumption is guaranted by a regularization term that we will introduce later. Figure 4 visualizes the differences between Euclidean and SNR distance. We easily notice that the SNR distance gives us more confidence in the determination whether a pair of samples is

similar or not. The Euclidean distance can be seen as an absolute error between the two samples by calculating only the distance from one point to another. On the other hand, the SNR distance can be seen as an relative error between the two samples by taking the distance to the origin of the feature space into account. Because the parameters of the

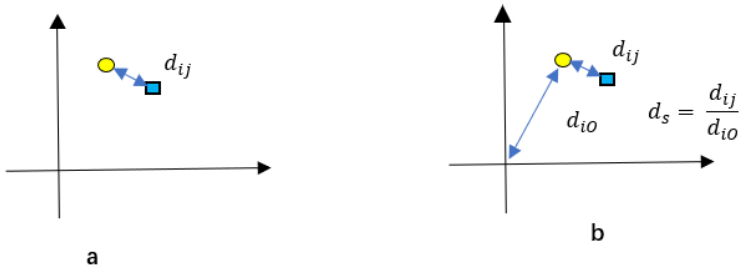


Figure 4: Illustration of Euclidean distance in a. Illustration of SNR distance in b.

neural network are optimized while training the learning function, we observe two forces that influence the mapping of the samples as shown in figure 5. The first force influences the intra-class distances of the samples. As we see in formula ..., by reducing the numerator $d_E(h_i, h_j)$, we also reduce the value of the SNR distance and the return of the objective function. Therefore the distance between samples of the same class decreases while training. The second force influences the inter-class disntaces. Because of the denominator $d_E(h_i)$, the distance to the mean of all samples increases regardless of the selected loss function. The pulling intra-classes-force and the pushing inter-classes-force effects a distribution in the feature space. Where samples of similar classes form close groups with far distance two the groups of other classes.

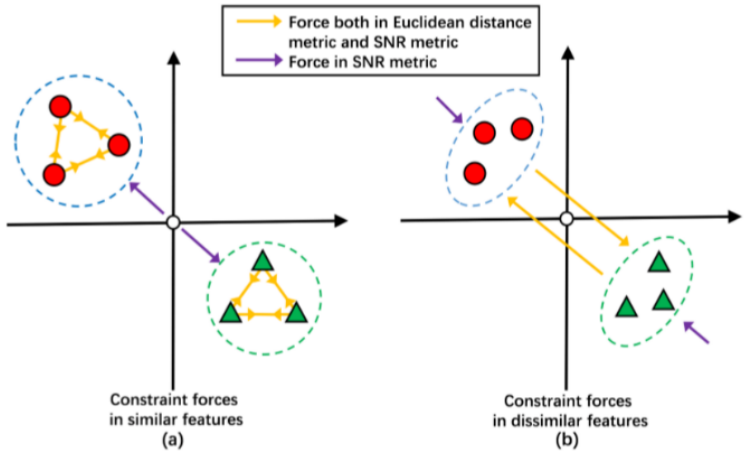


Figure 5: Illustration of the constraint force in Euclidean distance and SNR distance.

4.4 SNR Distance and Correlation

Now we derive the relationship between the SNR distance and the correlation coefficient of two samples in the feature space. Assuming a zero-mean-distribution in every component of the samples and an independence of the noise to the signal feature, we get

$$\begin{aligned}
\operatorname{corr}(h_i, h_j) &= \frac{\operatorname{cov}(h_i, h_j)}{\sqrt{\operatorname{var}(h_i)\operatorname{var}(h_j)}} = \frac{\mathrm{E}(h_i, h_j)}{\sqrt{\operatorname{var}(h_i)\operatorname{var}(h_j)}} && (11)\\
&= \frac{\mathrm{E}\left(h_i\left(h_i + n_{ij}\right)\right)}{\sqrt{\operatorname{var}(h_i)\operatorname{var}(h_i + n_{ij})}} = \frac{\mathrm{E}(h_i)^2}{\sqrt{\operatorname{var}(h_i)\operatorname{var}(h_i + n_{ij})}} && (12)\\
&= \frac{\operatorname{var}(h_i)}{\sqrt{\operatorname{var}(h_i)^2 + \operatorname{var}(h_i)\operatorname{var}(n_{ij})}} = \frac{1}{\sqrt{1 + \frac{\operatorname{var}(n_{ij})}{\operatorname{var}(h_i)}}} && (13)\\
&= \frac{1}{\sqrt{1 + \frac{1}{SNR_{ij}}}} = \frac{1}{\sqrt{1 + d_s(h_i, h_j)}}\,. && (14)
\end{aligned}$$

As we see in figure 6, the SNR distance is between two samples in the feature space, the greater the correlation coefficient between these two samples.

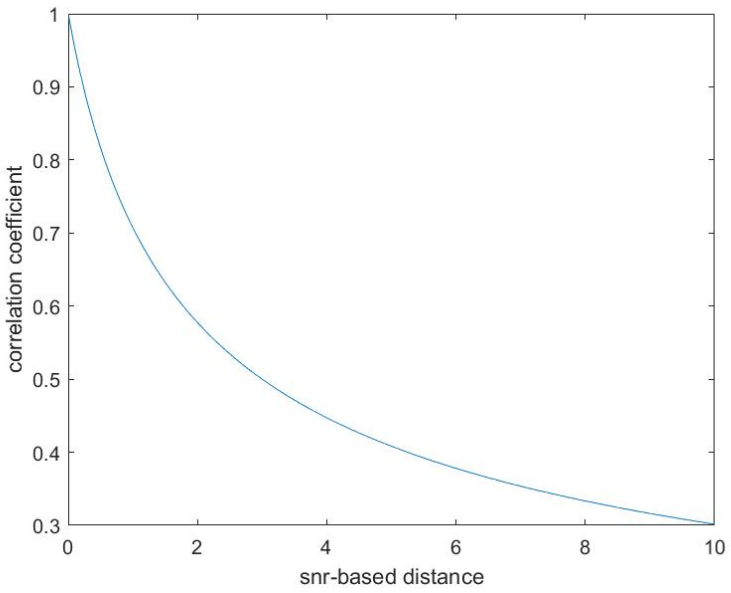


Figure 6: Illustration of the relationship between SNR distance metric and correlation coefficient of two samples.

4.5 Deep SNR-based Metric Learning

In this chapter we want to apply the SNR distance metric to various objective functions including contrastive loss, triplet loss, lifted structured loss and N-Pair loss. The learned

samples in the feature space are denoted as $h_i \in (h_1, ..., h_n)$. Given an anchor sample h_i, its positive sample is denoted as h_i^+ and its negative sample is denoted as h_i^-. We can formulate the SNR distance of two samples h_i and h_j in the feature space in the following objective functions as

$$d_{Sij} = d_S(h_i, h_j) = \frac{\mathrm{var}(h_i - h_j)}{\mathrm{var}(h_i)}\,. \tag{15}$$

To guarantee the zero-mean-distribution of the samples in the feature space, we add a regularization term λL_r to the objective functions. λ is a hyperparameter with a small value and L_r is defined as

$$L_r = \frac{1}{N}\sum_{i\in N} |\sum_{m=1}^{M} h_{im}|\,, \tag{16}$$

where N is the number of considered samples and M the dimension of the feature space.

4.5.1 DSML(cont)

Our DSML objective function for SNR-based contrastive embedding is

$$L = \sum_{i=1}^{N_i} d_S\left(h_i, h_i^+\right) + \sum_{j=1}^{N_j} \max\left(\alpha - d_S\left(h_j, h_j^-\right), 0\right) + \lambda L_r\,, \tag{17}$$

where N_i is the number of positive pairs and N_j is the number of negative pairs. α represents the margin for the negative pairs.

4.5.2 DSML(tri)

The objective function for SNR-based triplet embedding can be formulated as

$$L = \sum_{i=1}^{N} \max\left(d_s\left(h_i, h_i^+\right) - d_s\left(h_i, h_i^-\right) + \alpha, 0\right) + \lambda L_r\,. \tag{18}$$

4.6 DSML(lifted)

The SNR-based objective function for lifted loss is

$$L = \frac{1}{2N}\sum_{(i,j)\in P} \max(J_{ij}, 0) + \lambda L_r \tag{19}$$

$$\text{with} \quad J_{ij} = \max\left(\max_{(i,k)\in N}\left(\alpha - \beta d_{Sik}\right), \max_{(j,l)\in N}\left(\alpha - \beta d_{Sjk}\right)\right) + \beta d_{Sij}\,, \tag{20}$$

where P denotes the positive pairs and N denotes the negative pairs. α is the margin and β is a hyperparamter that ensures the convergence of the loss.

4.7 DSML(N-pair)

For every anchor x_i in N-pair loss, we consider a tuple $\{x_i, x_1^+, x_2^+, ..., x_N^+\}$ with x_i^+ as positive comparison sample and x_j where $j \neq i$ as negative samples. The idea of N-pair loss is not to calculate distances but to measure similarities which is done by a scalar product. As the SNR distance metric is not even a real metric from Mathematical point of view, we do not have a real scalar product in SNR based calculation. Because of that we have to introduce an SNR-based measure for simularity. Given two samples h_i and h_j in the feature space, we define their similarity S_ij as

$$S_{ij} = \frac{1}{d_{Sij}^2} = \mathrm{SNR}_{ij}^2 = \frac{\mathrm{var}(h_i)^2}{\mathrm{var}(h_i - h_j)^2}\,. \tag{21}$$

Accordingly, our SNR-based objective function for N-pair loss can be defined as

$$L = \frac{1}{N}\sum_{i=1}^{N} \log\left(1 + \sum_{j\neq i} \exp\left(S_{ij} - S_{ii}\right)\right) + \lambda L_r\,. \tag{22}$$

4.8 Deep SNR-based Hashing Learning

Hashing learning is to learn to encode the image samples to binary code. The binary codes of similar images should have short Hamming distances, and the binary codes of unmatched images have long Hamming distances.

To show the generality of SNR metric, authors deployed SNR distance metric to deep hashing learning and proposed Deep SNR-based Hashing methods (DSNRH). The main difference between deep hashing learning and deep metric learning is that the embeddings need to be quantized to binary features in hashing learning. And the similar labels are given as: if two images i and j share at least one label they are similar, otherwise, they are dissimilar.

5 Experiments of the paper

5.1 Experiments on Deep Metric Learning

5.1.1 Dataset

1. CARS196 dataset: contains 16185 images of 196 car models
2. CUB200-2011 dataset: includes 11788 images of 200 bird species
3. CIFAR10 dataset: contains 60000 32x32 color images of 10 classes. 100 images per class as the testing set, and the rest 59000 images as database set. From the database set, 500 images are randomly selected per class as training set.

5.1.2 Implementation Details and Evaluation Metrics

Using Tensorflow, authors built AlexNet for deep metric learning. They replace the fc8 with an embedding layer of M hidden units. And they choose the mini-batch stochastic gradient descent (SGD) with 0.9 momentum as optimizer, and fix the batch size of images as 100, except the N-pair method on CIFAR10 being set to 20.

For the clustering tasks, the authors make experiment on CUB200-2011 and CARS196, and use NMI and F1 score to measure the performance of different methods. For image retrieval tasks, they calculate the Recall@K for the experiment results on CUB200-2011 and CARS196, and record the MAP and F1 metric for the experiment results on CIFAR10.

5.1.3 Results and Analysis

Table 1 and table 2 show the performance of deep metric learning approaches on CARS196 and CUB200-2011. And we can observe that:

1. In comparison to deept metric learning combined with Euclidean distance, proposed SNR-based metric improves the performance on all bench-mark datasets.

2. Along the column, the best scores are all achieffed by DSML.

Table 1. Results on CARS196 with Alexnet.

Tasks	Image Clustering						Image Retrieval					
score (%)	F1			NMI			Recall@1			Recall@2		
embedding size	16	32	64	16	32	64	16	32	64	16	32	64
contrastive	9.2	10.6	11.0	31.5	34.4	33.3	8.9	14.0	16.3	10.3	16.1	18.4
DSML(cont)	12.9	11.9	11.8	39.9	37.0	36.1	15.1	16.5	18.0	17.5	18.6	201
triplet	19.4	16.9	15.4	50.9	47.9	46.8	24.8	20.6	19.5	28.2	23.5	22.1
DSML(tri)	25.6	**33.1**	**34.4**	52.5	**56.8**	**57.4**	**38.5**	**46.3**	**49.1**	**42.0**	**49.8**	**52.4**
lifted	27.1	29.0	28.1	53.1	54.4	53.9	37.2	39.1	40.6	41.2	42.9	44.3
DSML(lifted)	30.2	32.1	33.6	54.1	55.6	56.7	35.3	40.3	43.8	38.9	44.0	47.5
N-pair	26.9	29.9	29.5	51.8	53.5	53.6	32.9	36.3	38.3	36.7	39.8	42.1
DSML(N-pair)	**30.7**	**33.1**	32.7	**54.5**	54.4	56.4	37.8	40.4	44.9	39.8	44.5	48.6

Table 1: Results on CARS196 with Alexnet.

Table 3 shows the results of retrievbal tasks on CIFAR10 with two retrieval strategies: Euclidean ranking and Hamming ranking. Similar to table 1 and table 2, DSML outperforms the related Euclidean-based metric learning methods.

Figure 7 shows the t-SNR visualization of the features learned by DSML(cont) and contrastive on CIFAR10. It is obvious that the features learned by DSML(cont) have more clear boundary and discriminative structures, while contrastive loss learned a vague structure. Despite the fact that the assumption on which the SNR distance metric based is not obviously established, the experiment result clearly shows that the SNR distance metric is more powerful than the Euclidean distance metric.

Table 2. Results on CUB200-2011 with Alexnet.

Tasks	Image Clustering						Image Retrieval					
score(%)	F1			NMI			Recall@1			Recall@2		
embedding size	16	32	64	16	32	64	16	32	64	16	32	64
contrastive	14.6	18.7	19.3	41.6	46.6	47.4	15.8	25.7	29.7	18.0	28.6	32.7
DSML(cont)	19.6	19.7	22.7	47.5	47.8	50.5	22.2	27.2	33.1	25.3	30.6	36.4
triplet	23.6	22.1	21.7	56.5	55.6	55.3	33.9	32.8	32.6	37.8	36.4	35.6
DSML(tri)	36.1	39.0	40.3	63.0	64.0	**65.6**	45.7	**49.8**	**51.6**	49.3	**53.5**	**54.9**
lifted	36.0	36.5	37.2	60.9	61.1	61.4	43.2	44.5	46.8	46.4	47.8	50.4
DSML(lifted)	**41.3**	**43.9**	**45.8**	**63.5**	**64.5**	65.4	**46.0**	48.8	51.0	**49.4**	51.9	54.4
N-pair	34.7	35.7	37.6	59.6	60.0	61.5	39.9	40.7	43.1	43.3	44.4	46.9
DSML(N-pair)	37.6	38.1	40.5	62.4	61.9	63.1	42.3	46.2	48.5	48.6	49.7	51.9

Table 2: Results on CUB200-2011 with Alexnet.

Table 3. Retrieval results on CIFAR10 with AlexNet.

	Euclidean Ranking						Hamming Ranking					
score (%)	MAP@59000			F1@5000			MAP@59000			F1@5000		
embedding size	16	32	64	16	32	64	16	32	64	16	32	64
contrastive	75.5	73.4	69.3	69.1	67.2	61.4	65.5	66.9	61.8	61.2	62.2	56.9
DSML(cont)	**80.0**	**79.8**	**79.0**	**72.9**	**72.7**	**72.1**	**73.7**	**76.6**	**76.9**	**70.0**	**72.2**	**71.4**
triplet	75.9	77.3	75.8	70.7	71.2	70.3	71.9	73.7	74.3	67.3	70.2	69.8
DSML(tri)	78.4	78.3	77.4	72.4	72.5	71.6	73.4	74.5	75.3	69.9	70.8	70.8
lifted	63.7	54.6	55.5	60.6	52.0	52.0	60.3	52.1	53.9	54.9	50.0	50.8
DSML(lifted)	78.1	76.2	76.7	73.5	71.1	71.8	66.9	74.3	70.7	58.1	70.5	67.1
N-pair	53.5	51.1	39.5	49.5	47.5	37.8	48.4	48.9	38.6	45.9	46.4	37.3
DSML(N-pair)	62.1	64.1	56.6	57.1	58.8	52.1	55.2	62.0	53.6	50.2	57.3	49.6

Table 3: Retrieval results on CIFAR10 with Alexnet.

5.2 Experiment on Hashing Learning

5.2.1 Datasets

1. CIFAR10: 1000 images per class as the test query set, and the rest images are selected as training set and database set.

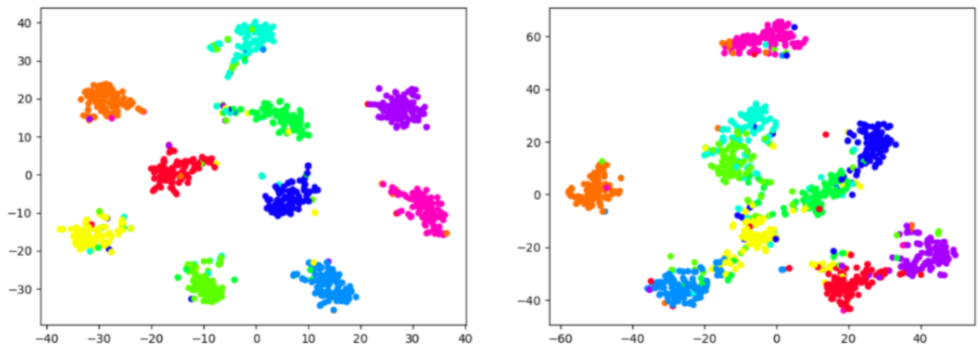
2. NUS-WIDE: consisted of 269,648 images associated with 81 tags. 100 images per class as the test query images, and the rest are used as the training set and database set.

5.2.2 Implementation Details and Evaluation Metrics

In DSNRH, CNN-F network architecture is deployed. Optimizer is mini-batch stochastic gradient descent with 0.9 momentum. Results are evaluated by MAP@5000.

5.2.3 Results

Table 4 shows the performance of proposed DSNRH and other five deep hashing methods, including DPSH, DTSH, DRSCH, DSCH.DDRH. DSNRH has the best performance



(a) DSML(cont) (b) contrastive

Figure 7: The t-SNR visualization of features learned by DSML(cont) method and the contrastive method with Euclidean distance on the CIFAR10 dataset.

among other methods. Though DPSH and DTSH are based on the network of the same architecture, their performance is still about 2 % and about 12 % less than that of DSNRH on CIFAR10 dataset and NUS-WIDE dataset respectively.

method	CIFAR10			
	16 bits	24 bits	32 bits	48 bits
DSRH [44]	0.608	0.611	0.617	0.618
DSCH [43]	0.609	0.613	0.617	0.620
DRSCH [43]	0.615	0.622	0.629	0.631
DTSH [35]	0.915	0.923	0.925	0.926
DPSH* [14]	0.903	0.885	0.915	0.911
DSNRH(Ours)	**0.925**	**0.932**	**0.934**	**0.940**

(a)

method	NUS-WIDE			
	16 bits	24 bits	32 bits	48 bits
DSRH [44]	0.609	0.618	0.621	0.631
DSCH [43]	0.592	0.597	0.611	0.609
DRSCH [43]	0.618	0.622	0.623	0.628
DTSH [35]	0.756	0.776	0.785	0.799
DPSH [14]	0.715	0.722	0.736	0.741
DSNRH(Ours)	**0.830**	**0.840**	**0.852**	**0.862**

(b)

Table 4: Deep hashing methods on CIFAR10 and NUS-WIDE.

5.3 Discussion

The experiment results on benchmark datasets show that the SNR metric can boost the performance of deep metric learning because it can jointly pull features of the similar samples closer and enlarge the distance of features of inter-class samples. And the results also indicate the generalization of SNR metric in deep hashing learning. In addition, if authors could adopt another state-of-art network like ResNet instead of AlexNet to execute the experiments, the results would be more persuasive. Since there is not only one metric, Euclidean distance, authors should make a comprehensive comparison by taking other distance metrics into comparison. Last but not least, as we have mentioned above, this SNR metric is based on the assumption that all the components of features, no matter from which class, should follow the same distribution. From the results, it

seems that SNR metric works pretty well so the assumption may be probably established. Here I will try to infer the distribution that h_i may follow. h_i is the sum of previous layer output, which can be formulated as:

$$h_i = \sum_{k=1} X_k \,, \tag{23}$$

where h_i denotes one of the components of features, and X_k is the output value of previous layer. X_k must obey a certain distribution which is most likely the quasi Gaussian distribution because of the Hierarchical structure of neural networks. According to the Liapunov law, we can deduce that every component of h_i should also follow quasi Gaussian distribution. Notice that authors add a regularization to the loss function which can not only constrain the mean of h_i to be very close to zero but also lead to a very small standard deviation of h_i. Therefore, the distribution of h_i would be a quasi-Gaussian distribution with a mean very close to zero and small standard deviation. Since the mean and the standard deviation is very small, the difference between mean and standard deviation of different h_i are even smaller so that all h_i approximately follow the same distribution. This could well explain why the SNR metric is so successful.

5.4 Conclusion

In this paper, authors proposed a distance metric based on Signal-to-Noise Ratio (SNR), which can better present discriminative features and preserve the correlation coefficient of sample pairs than the Euclidean distance. By simply replacing Euclidean distance with SNR distance, the authors construct deep SNR-based metric learning which shows its superiority to state-of-the-art deep metric learning methods on three benchmarks. As an extension, the authors also deployed SNR metric to hashing learning and the proposed deep SNR-based hashing learning methods achieve an outstanding performance on two benchmarks. Though it is not that persuasive because of the networks used in the experiment and lacking a comprehensive comparison with other more distance metrics, we think at least that the SNR-based distance is a good replacement for the Euclidean distance.

6 Network Architectures

While the experiments in the paper are based on AlexNet, our own experiments are based both on AlexNet and on ResNet.

6.1 AlexNet

AlexNet is a convolutional neural network architecture which was introduced in the paper "ImageNet Classification with Deep Convolutional Neural Networks" in 2012. AlexNet consists of eight layers. The first five layers are all convolutional layers. In contrast to other convolutional neural network architectures of this time, AlexNet works with

overlapping pooling. This means that if a pooling layer consists of a grid of pooling units spaced s pixels apart and the size of the summarized neighborhood centered at the pooling units is of size $z \times z$, then z has to be greater than s. In case of AlexNet, that is true by choosing $s = 2$ and $z = 3$. The last three layers are fully-connected layers. AlexNet uses rectified linear units as output layers.

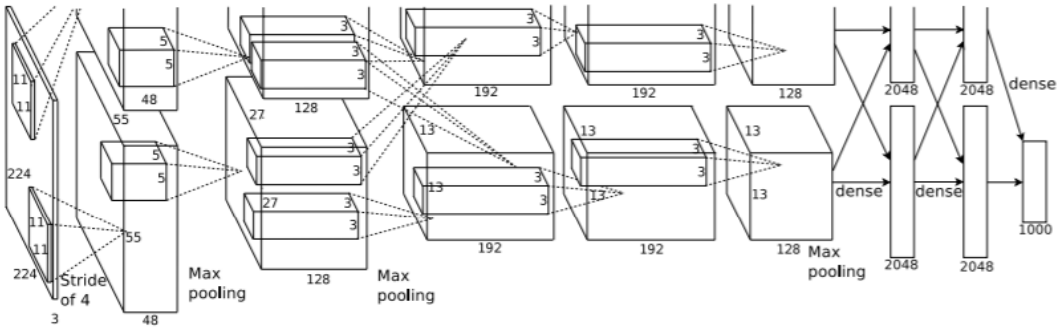


Figure 8: All layers of AlexNet.

6.2 ResNet

ResNet is a neural network architecture which was introduced in the paper "Deep Residual Learning for Image Recognition" in 2015. The basic idea of ResNet is to skip layers in the training process. This is done by so called shortcut connections that deliver an input to a layer which is the activation output of the layer two positions before. For high numbers of layers, ResNet architectures show much better results than AlexNet on benchmarks like CIFAR10.

7 Mahalanobis-Metric

7.1 Definition

A function $d: X \times X \to \mathbb{R}_0^+$, where X is a vector space, is defined as metric, if d satisfies the following conditions:

1. triangular inequality:

$$d(x_i, x_j) + d(x_j, x_k) \geq d(x_i, x_k) \quad \forall x_i, x_j, x_k \in X \tag{24}$$

2. non-negativity:

$$d(x_i, x_j) \geq 0 \quad \forall x_i, x_j \in X \tag{25}$$

3. symmetry:

$$d(x_i, x_j) = d(x_j, x_i) \quad \forall x_i, x_j \in X \tag{26}$$

4. distinguishability:

$$d(x_i, x_j) = 0 \quad \Leftrightarrow \quad x_i = x_j \quad \forall x_i, x_j \in X \tag{27}$$

By satisfying all four conditions, the Euclidean distance is a metric. The SNR distance is not a metric, because it is obviously not symmetric for example. If a function satisfies the first three conditions, we call the function pseudometric.

Now, we define a function d_L that applies a linear transformation to the input vectors and returns the squared euclidean distance:

$$d_L(x_i, x_j) = \|L(x_i - x_j)\|_2^2 \quad \forall x_i, x_j \in X\,, \tag{28}$$

where L is the linear transformation. If L is a matrix of full rank, d_L is a metric. Otherwise, d_L is a pseudometric.

As we define M as

$$M = L^T L\,, \tag{29}$$

M is a positive semidefinite matrix for every real-valued matrix L. We define the Mahalanobis metric d_M as

$$d_M = (x_i - x_j)^T M (x_i - x_j) \quad \forall x_i, x_j \in X\,. \tag{30}$$

In case of a full-rank-matrix M, the Mahalanobis metric is a real metric from Mathematical point of view. In any case, the Mahalanobis metric is a pseudometric.

7.2 Training the Metric

The Mahalanobis Metric can be chosen as a distance measure in the loss function and has to be optimized while training the network. The entries of matrix M can be considered as hyperparameters of the network. One common approach for optimizing the metric is to select those entry values for M that increase the samples of the same class in the k-nearest neighborhood. There are also some helpful approaches from Relevant Component Analysis and Convex Optimization to optimize the metric while training.

7.3 Relative Mahalanobis Distance

As proven before, the SNR metric can be considered as a relative Euclidean metric, because in case of a zero-mean-distribution, the SNR metric d_S can be written as

$$d_S(h_i, h_j) = \frac{d_E(h_i, h_j)^2}{d_E(h_i, h_i)^2}\,, \tag{31}$$

where d_E is the Euclidean distance. The authors of the paper explain the superiority of the SNR distance in their experiments by this fact amongst other things. Therefore,

we define our own relative Mahalanobis metric d_{rM} analogous to the relative Euclidean metric as

$$d_{rM} = \frac{d_M(h_i, h_j)}{d_M(h_i, h_i)}, \tag{32}$$

where d_M is the Mahalanobis metric. If M equals the identity matrix, the Mahalanobis metric already corresponds to the squared Euclidean metric. Thus, we do not use squared metrics in the relative Mahalanobis metric, even though we use squared metrics in the relative Euclidean metric.

8 Our Experiments

8.1 Dataset

Our dataset is CARS196. It consists of 16,185 images of 196 different car models. The training set contains 8,144 images, the test set contains 8,041 images.

8.2 Implementation Details

We used pytorch to implement the neural networks and chose the architecture of Confusezius as basic framework for the training part. The architecture delivers dataloader functions that allow to train the models in customized image batches. Because the loss functions need different batch shapes in the training, we use the preimplemented sampler functions to organize the batches for the loss functions. The loss functions have the metric name as an additional parameter. That allows us to train the same loss function with different metrics. The training is based on *demo.py* that can be customized via command line parameters. We trained models in AlexNet and ResNet50 architecture. For every architecture we trained with the loss functions triplet loss and lifted loss and trained for all five metrics Euclidean distance d_E, SNR distance d_S, relative Euclidean distance d_{rE}, Mahalanobis distance d_M and relative Mahalanobis distance d_{rM}.

The evaluation is based on *evaluate.py* that can also be customized via command line parameters. We implemented Recall@K, MAP@K and F1@K but only calculated Recall@1 and Recall@2 for the comparison to the results from the paper. Usually the recall is determined by calculating the ratio of true positives to true postives and false negatives. Since we do not use our network for direct classifiation, but just project our input data into a multidimensional feature space, we choose a k-nearest-neighbors approach from the paper to calculate the recall. In this approach, Recall@K is calculated by that an anchor scores one in case of a similar sample in the K-nearest neighborhood. Otherwise, the anchor scores zero. We determine the Recall@K by calculating the arithmetic mean of these scores.

8.3 Results

At first we started with the repitition of the experiments from the paper. Table 5 shows the results from AlexNet with a 16-dimensional feature space for Recall@1 and Recall@2. Like in the paper, our best combination of loss function and metric for AlexNet in Re-

loss function	metric	Recall@1	Recall@2
triplet	Euclidean	23.1	34.9
triplet	relative Euclidean	29.4	45.7
triplet	SNR	31.1	43.2
lifted	Euclidean	26.0	39.4
lifted	relative Euclidean	26.0	40.1
lifted	SNR	30.3	41.7

blabla

call@1 evaluation is triplet loss with the SNR metric. In Recall@2 evaluation, we have the relative Euclidean with the best score which is only 2.5 % better than the SNR metric. In triplet loss, the relative Euclidean metric and the SNR metric have similar scores what supports the hypothesis that the relative Euclidean metric is an approximation of the SNR metric. Just in in lifted loss with Recall@1 evaluation, we see 4.3 % difference between SNR and relative Euclidean. Whlie we see strong superiority of SNR in triplet loss and similarity in lifted loss in the paper, our experiments show weak superiority of SNR in both triplet loss and lifted loss.

Table 6 shows the results for ResNet50 with a 16-dimensional feature space for Recall@1 and Recall@2. Triplet loss with Euclidean metric gets the highest score in Recall@1

loss function	metric	Recall@1	Recall@2
triplet	Euclidean	65.6	77.0
triplet	relative Euclidean	60.7	73.5
triplet	SNR	58.5	69.0
triplet	Mahalanobis	63.9	73.5
triplet	relative Mahalanobis	63.1	72.1
lifted	Euclidean	64.6	77.9
lifted	relative Euclidean	43.4	58.5
lifted	SNR	43.9	57.0
lifted	Mahalanobis	53.9	66.7
lifted	relative Mahalanobis	33.3	49.0

blabla

evaluation and lifted loss with Euclidean metric gets the highest score in Recall@2 evaluation. Both in triplet loss and lifted loss, the Mahalanobis metric is the second best

metric behind the Euclidean metric. The relative Mahalanobis metric has in triplet loss an average performance, but gets the worst score in lifted loss. As expected, all evaluation scores from ResNet50 are better than the evaluation scores from AlexNet.

8.4 Discussion

All in all our experiments for AlexNet support the results from the paper that the SNR metric with triplet loss gets the best evaluation results, even if the superiority in our experiments was not as strong as in the paper. Apart from training AlexNet with lifted loss, our experiments both with AlexNet and ResNet50 verified the hypothesis that the relative Euclidean metric can be seen as an approximation of the SNR metric. The Mahalanobis metric was the second best metric for both loss functions, while the relative Mahalanobis metric had an anverage performance in triplet loss and the worst performance in lifted loss. In contrast to AlexNet, the SNR metric could not show superiority in the evaluations for ResNet50. But we also see in the paper that the SNR score improves by increasing the number of feature dimensions while the Euclidean score descreases by increasing the number of feature dimensions. Because of complexity and runtime reasons, we only trained model with 16 dimensional feature spaces. A possible superiority of SNR in ResNet50 for higher feature dimensions has still to be checked.

9 Conclusion

In this report, we introduced two different different training approaches in deep metric learning. While structure learning focusses on the optimization of loss functions and data sampling, distance learning tries to improve the distance measures in the feature space. We introduced the basic concepts of the paper "A Robust Distance Metric for Deep Metric Learning" and summarized the experimental results of the paper. Based on the idea that the SNR metric can be considered as a relative Euclidean metric, we designed the relative Mahalanobis metric which could not show superiority in our own experiments. We could verify the superiority of the SNR metric in our experiments with AlexNets. In our experiments with ResNet50, the Euclidean metric got the highest score for both loss functions.

References

[1] Vorname1 Nachname1 Titel1 Verlag1, Jahr1, Ort1

[2] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research, 10(Feb):207–244, 2009.

[3] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In null, pages 1735–1742. IEEE, 2006.

[4] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In CVPR, pages 815–823, 2015.