This WACV 2023 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

InDiReCT: Language-Guided Zero-Shot Deep Metric Learning for Images

Konstantin Kobs Michael Steininger Andreas Hotho
University of Würzburg
Am Hubland, 97074 Würzburg
{kobs,steininger,hotho}@informatik.uni-wuerzburg.de

Abstract

*Common Deep Metric Learning (DML) datasets specify only one notion of similarity, e.g., two images in the Cars196 dataset are deemed similar if they show the same car model. We argue that depending on the application, users of image retrieval systems have different and changing similarity notions that should be incorporated as easily as possible. Therefore, we present Language-Guided Zero-Shot Deep Metric Learning (LanZ-DML) as a new DML setting in which users control the properties that should be important for image representations without training data by only using natural language. To this end, we propose InDiReCT (**I**mage representatio**n**s using **Di**mensionality **Re**duction on **C**LIP embedded **T**exts), a model for LanZ-DML on images that exclusively uses a few text prompts for training. InDiReCT utilizes CLIP as a fixed feature extractor for images and texts and transfers the variation in text prompt embeddings to the image embedding space. Extensive experiments on five datasets and overall thirteen similarity notions show that, despite not seeing any images during training, InDiReCT performs better than strong baselines and approaches the performance of fully-supervised models. An analysis reveals that InDiReCT learns to focus on regions of the image that correlate with the desired similarity notion, which makes it a fast to train and easy to use method to create custom embedding spaces only using natural language.*

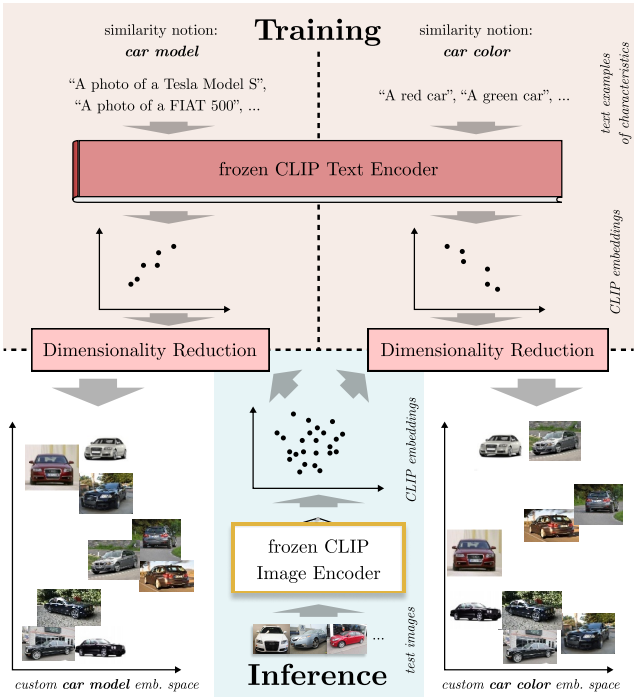

Figure 1. During InDiReCT's training (top half), different aspects of the desired similarity notion (e.g. car model or color) are collected in form of text prompts. CLIP's frozen text encoder embeds them and a dimensionality reduction method is learned to extract the dimensions that encode the similarity notion's aspects. During inference (bottom half), the trained dimensionality reduction is applied to CLIP encoded images to obtain custom image embeddings representing the desired similarity notion.

1. Introduction

Deep Metric Learning (DML) is the task of training deep neural networks that map input items to a low-dimensional manifold such that similar items are represented by vectors close to each other [22, 11]. In the usual DML setting, training examples are needed that let the model learn which image properties make an image pair (dis)similar. For example, in the Cars196 dataset's setting [15], two images are deemed similar if they show the same car model. Factors such as the car color, its orientation, or the image's environment should be suppressed by the embedding process. Other datasets have different image properties to define when two images are similar. We call this high-level interpretation of when two inputs are deemed similar a **similarity notion**. For Cars196, the similarity notion is "Two car images are similar if they show the same car model". During testing, the ability of the neural network to generalize this learned similarity notion to new unseen classes (e.g. new car models) is measured.

Often, people have different similarity notions depending

on the task at hand or personal preference. It is thus desirable to be able to quickly adapt to changing similarity notions. However, large labeled training datasets are needed to train a model for a new similarity notion, which is time-consuming and tedious for users to create. We thus aim for a zero-shot setting, where no training images and labels are needed. We argue that users often can express the desired similarity notion using words, e.g., "Two car images are similar if both cars have the same color". This is especially the case when there are categorical aspects with names that sort the images into disjoint classes. More specifically, users can list a set of distinct aspects describing the similarity notion, e.g. "a red car", "a green car", ... The use of language simplifies the process of expressing custom similarity notions, which alleviates the problem of collecting new labeled datasets.

As a first contribution of this work, we introduce a new task called **Language-Guided Zero-Shot Deep Metric Learning (LanZ-DML)**: Given a set of images $\mathcal{I}$ and a desired similarity notion $\mathcal{S}$ that is described using text $\mathcal{T}_\mathcal{S}$. Train a Deep Metric Learning model using only the text input $\mathcal{T}_\mathcal{S}$ such that the resulting model can embed images $\mathcal{I} \rightarrow \mathbb{R}^n$ to n-dimensional embedding vectors, making image embeddings more similar if they are deemed similar regarding the similarity notion. For optimization, no training images or labels are allowed (thus zero-shot).

Our second contribution is InDiReCT (**I**mage representatio**n**s using **Di**mensionality **Re**duction on **C**LIP embedded **T**exts), a model for LanZ-DML that uses a list of text prompts as input and learns a transformation that maps images to a vector space that reflects the desired similarity notion. It utilizes the Contrastive Language-Image Pretraining (CLIP) [26] model as a static general purpose feature extractor for images and texts. We assume that CLIP embeddings for images and texts encode similar concepts in similar directions of the embedding space and that image descriptions can focus on certain properties. For example, the text description "a photo of a red car" focuses on the car color and not on other features, such as the car's position, orientation, or environmental factors.

Figure 1 gives an overview of InDiReCT. During training, CLIP's *fixed* text encoder represents different characteristics of a desired similarity notion $\mathcal{S}$ as 512-dimensional vectors, e.g. "a red car", "a white car", and so on to encode the car color. We then extract the largest variations of these vectors in the embedding space by applying a dimensionality reduction method to these text representations, focusing on the changing aspects and abstracting away other non-related dimensions. Learning the dimensionality reduction is fast and often, only a few dozen text prompts are needed. Also, no training images or labels are used, only text prompts.

During inference, images are fed through CLIP's *fixed* image encoder and the trained dimensionality reduction. Assuming that CLIP's embeddings encode similar concepts in similar embedding space directions for both modalities, the resulting image representations are focused on the same dimensions as described by the text prompts. Finally, lower-dimensional vectors can be used to find images similar to an image w.r.t. the desired similarity notion.

For our third contribution, we provide experimental evidence on five datasets and overall thirteen similarity notions, i.e. different properties by which the image embeddings should vary. We show that InDiReCT consistently achieves better performance in retrieving similar images w.r.t. the desired similarity notion than strong baselines in the zero-shot setting and even approaches fully supervised baselines. We also analyze the influence of changing embedding sizes, CLIP model sizes, and number of text prompts, and visualize the image regions InDiReCT focuses on to create image representations. Our qualitative analysis shows that InDiReCT pays attention to pixels that are important in identifying the desired similarity notion. Our code is publicly available.[1]

2. Related Work

Deep Metric Learning (DML) aims to learn neural networks that map input items to low-dimensional vectors such that similar items are close together in the embedding space [22]. In this work, we focus on images as items, which can be used for image retrieval [33], face recognition [6], and image clustering [11]. Usually, the model is trained on images, organized into classes, so binary similarity annotations are readily available for each pair of data points [11]. Testing then uses a disjoint set of image classes to measure the model's generalization ability, but the data is semantically similar to the training data, e.g. Cars196 [15] only shows cars and face recognition datasets [6] contain faces.

Studying DML generalization for images outside of the training domain has recently become popular [20, 28, 19, 8, 7, 32]. However, all proposed methods to improve the generalization performance to new datasets still use training images. In our setting, no training images are allowed, but only text prompts to create an embedding space specifically tailored to a desired similarity notion. For this, we use a fixed CLIP [26] model to extract general purpose features.

The ability to rank possible text labels for an image using the cosine similarity of CLIP embeddings has been used in the original paper to perform zero-shot image classification [26]. For classification, the class names need to be known during inference while in LanZ-DML, we create image embedding spaces reflecting the desired similarity notion. Hence, the model needs to be able to handle images of unknown objects and characteristics, e.g. new car models.

Baldrati et al. use CLIP to alter fashion image embeddings using text prompts [1], e.g. the image of a black dress is combined with the text "is red" to find images of red

[1] https://github.com/LSX-UniWue/InDiReCT

dresses. While exploiting similar properties of CLIP, we only use text prompts for training a transformation to focus on the desired similarity notion. Image retrieval also uses joint text-image embeddings search for image contents using text [18, 2]. We use text exclusively during training, not during inference. To the best of our knowledge, no other work has a comparable task setting or method as our paper.

3. Methodology

We now introduce InDiReCT, our method for Language-Guided Zero-Shot Deep Metric Learning on images. It makes use of a fixed CLIP [26] as a general-purpose feature extractor for images and texts, which encodes similar concepts in both modalities to similar embedding directions. CLIP consists of encoders for image and text. It is pretrained on 400 million image-text pairs, optimized to embed both images and text such that embedding vectors of corresponding images and texts are more similar than different image-text pairs. Similarity is measured using cosine similarity, i.e. the cosine of the angle between two vectors. Due to the training task, CLIP learns to extract broad image features that can be correlated with/expressed by language. Intuitively, we aim to learn a transformation that focuses on the most important features extracted by CLIP regarding the desired similarity notion. Figure 1 shows InDiReCT's training and inference.

3.1. Training

In the training phase, n different text prompts are created that describe certain characteristics of the desired similarity notion. For example, if the target images show cars and we want to differentiate them by their color, we create a list of texts $\mathcal{T}_S$ such as "a red car", "a blue car", "a white car", and so on. The text prompts should only vary in the notion that we want to differentiate (here, the color descriptions). Note that the aspects in the training text prompts are chosen independently from inference data, since inference labels are not known during training and we want to generalize to new aspects of the similarity notion as well.

When feeding all texts through CLIP's text encoder, the resulting r-dimensional vectors[2] $t_i \in \mathbb{R}^{1\times r}$ for $i \in \{1,\dots,n\}$ vary in certain directions. This is introduced by the change in aspects of the desired similarity notion in the text prompts. Here, the variation of the vectors is only explained by the change of color names in the texts.

Due to CLIP encoding similar concepts to similar embedding dimensions, varying the same aspects in images and texts should result in embeddings that vary in similar directions. Our goal is to find these directions using the text embeddings and suppress all other directions in the image embedding space, which are influenced by undesired factors. Given the n text representations $t_i \in \mathbb{R}^{1\times r}$ for $i \in 1,\dots,n$, we thus aim to identify the dimensions that vary the most in order to learn a transformation that retains these directions while reducing the embedding to r' dimensions (similar to dimensionality reduction techniques such as PCA [31]). For this, we transform the text representations t_i using a matrix $U \in \mathbb{R}^{r\times r'}$ and reconstruct them using $U^\top$. We optimize U with gradient descent to minimize reconstruction loss L:

$$t_i^{\text{norm}} = \frac{t_i}{\|t_i\|} \quad (1) \qquad t_i^{\text{recon}} = \frac{t_i' U^\top}{\|t_i' U^\top\|} \quad (3)$$

$$t_i' = \frac{t_i^{\text{norm}} U}{\|t_i^{\text{norm}} U\|} \quad (2) \qquad L = \frac{1}{n}\sum_{i=1}^{n} \arccos(t_i^{\text{norm}} t_i^{\text{recon}\top}) . \quad (4)$$

CLIP's use of cosine similarity as a similarity measure for embeddings disregards the length of all vectors, so we map input vectors t_i and their reconstructions to a unit hypersphere (Equations (1) and (3)). Then we minimize the mean spherical distance (Equation (4)) between the input and reconstructed vectors [12]. It is the distance between the vectors along the surface of the hypersphere, scaling linearly with the vector angle. The training objective effectively minimizes the angles between the inputs and reconstructions.

In addition, the lower-dimensional embedding projections t_i' are also mapped to a unit hypersphere (Equation (2)). This ensures that the reconstruction only uses the angles between the r'-dimensional vectors, keeping cosine similarity as a similarity measure in the lower-dimensional space, while preserving the varying directions of the text embeddings.

Since only up to a few hundred text prompts are used and only the matrix U must be optimized, L typically converges really fast. The whole optimization process usually finishes in less than a minute on a common laptop's CPU, allowing InDiReCT to adapt to new similarity notions fast.

3.2. Inference

Given query and reference images, we feed them through CLIP's fixed image encoder and apply the learned transformation to map these embeddings $v_i \in \mathbb{R}^{1\times r}$ ($i \in \{1,\dots,m\}$) to r' dimensions on a unit hypersphere:

$$v_i^{\text{norm}} = \frac{v_i}{\|v_i\|} \quad (5) \qquad v_i' = \frac{v_i^{\text{norm}} U}{\|v_i^{\text{norm}} U\|} . \quad (6)$$

These vectors can be compared using the cosine/dot product similarity to find similar images w.r.t. the desired similarity notion. Since the transformation learns to suppress dimensions that do not vary in the text prompts, these dimensions are also suppressed for images, e.g., a car *model* dimension in the CLIP embedding space is suppressed when training with the similarity notion "car color".

4. Experiments

We now perform multiple experiments using InDiReCT and other baselines. Since we are in a zero-shot learning

[2] $r = 512$ for CLIP's base model, but it is not limited to that number

setting, we have no access to labeled training images. Hyperparameters cannot be tuned on a validation dataset, since labeled data is not allowed. We thus define prompts and set hyperparameters based on commonly used values or educated guesses. This resembles the real world scenario, where users do not have any training data at hand to verify and optimize their input to the system.

We implement InDiReCT using PyTorch [23] and sample the initial values of U from $\mathcal{N}(0, 0.1)$. We then optimize U using Adam [13] with a learning rate of 0.01 until it does not improve the loss L (Equation (4)) for 100 consecutive iterations. We reduce CLIP's vectors to 128 dimensions, which is a common embedding size for DML models [22].

We train models and compute image embeddings for each dataset and similarity notion. We follow the standard evaluation setting of DML and measure the retrieval performance for these embeddings using the Mean Average Precision at R (MAP@R) and Precision at 1 (Prec@1) [22]. Results for other evaluation metrics are in the appendix.

4.1. Datasets and Similarity Notions

We experiment with five datasets and overall thirteen similarity notions, which are listed in Table 1. For each dataset, we define one to four similarity notions, e.g. the "Car Model" similarity notion of the Synthetic Cars [14] and Cars196 [15] datasets can be expressed as "Two car images are similar if they show the same car model". Other notions can be formulated accordingly. Given a similarity notion, the datasets are split into different numbers of test classes (shown in the "Class Count" column), e.g. we use the 98 car models from Cars196's test dataset. We create multiple text prompts for each similarity notion by collecting possible aspects and inserting them into a prompt template (listed in the corresponding columns). The varying aspects are collected from different sources, such as an online car dealer website ("Car Model" and "Manufacturer"), the CSS2.1 color names ("Car Color" and "Background Color"), or the dataset's training data's labels (e.g. "Bird Species"). This promotes text prompts being possibly different from the test class labels, ensuring a realistic DML scenario, where train and test classes are commonly disjoint. More details on datasets, similarity notions, and prompts are in the appendix.

4.2. Baselines

InDiReCT is the first method for Language-Guided Zero-Shot Deep Metric Learning, i.e. it can efficiently generate specialized embedding spaces for images based on the desired similarity notions. Visualizations of embeddings produced by InDiReCT for similarity notions of the Cars196 dataset can be found in the appendix. Since InDiReCT does not use any training images, it is not fair to compare it to fully supervised baselines. However, we still contrast some of InDiReCT's results with fully supervised models and an Oracle baseline. We use the following methods in our experiments to get a sense of how well InDiReCT performs.

Random Baseline For this baseline, we sample r'-dimensional embedding vectors for each image uniformly from the unit hypersphere [5]. This baseline indicates the performance lower bound for all methods.

CLIP [26] This baseline feeds all images through CLIP's image encoder and uses the unmodified r-dimensional vectors as embeddings ($r = 512$). Due to the broad set of features CLIP extracts, its performance should already be quite good. However, since it does not focus on specific dimensions, InDiReCT is assumed to perform better while having fewer dimensions. Even more so, CLIP cannot adapt its embeddings based on the desired similarity notion, i.e., it always yields the same embeddings for an image. This limitation holds for all embedding methods that do not use additional data regarding the desired similarity notion.

Random Transformation InDiReCT optimizes a transformation that is applied to CLIP's image embeddings to achieve an embedding specialized towards a similarity notion expressed by text. We evaluate how well the learning procedure of InDiReCT improves the performance by leaving U as initialized for testing, i.e. sampled from $\mathcal{N}(0, 0.1)$. We hypothesize that this baseline should, on average, be worse than both InDiReCT and the CLIP baseline.

Principal Component Analysis (PCA) [31] PCA is a popular dimensionality reduction technique which finds orthogonal directions that explain the largest variation in the data. We test it as a possible alternative to our proposed method. In contrast to our method, PCA solves for principal components analytically, requiring r' to be strictly smaller than the number of input data points [24]. This is not satisfied for almost all scenarios in our experiments, since we only use a few text prompts while wanting to reduce CLIP's embeddings to a target size of 128 dimensions. We thus can apply PCA only on the datasets that we collect more than 128 text prompts for, i.e. the "Car Model" similarity notion for the Synthetic Cars and Cars196 datasets.

Linear Autoencoder (LAE) The LAE is an alternative to PCA that provably spans the same subspace while being able to be trained using gradient descent [25]. Formally, we optimize the weight matrices $W_1 \in \mathbb{R}^{r \times r'}$, $W_2 \in \mathbb{R}^{r' \times r}$ and bias vectors $b_1 \in \mathbb{R}^{1 \times r'}$, $b_2 \in \mathbb{R}^{1 \times r}$ with Adam (learning rate 0.01 and early stopping after 100 iterations) to minimize the loss function $L_{LAE} = \sum_{i=1}^{n} \sum_j ((t_i^{\text{norm}})_j - (W_2(W_1 t_i^{\text{norm}} + b_1) + b_2)_j)^2$. Image vectors are then transformed with $v_i' = W_1 v_i^{\text{norm}} + b_1$.

Table 1. Details on the datasets and similarity notions used for our experiments.

Dataset	Similarity Notion	Class Count	Prompt Template	Aspects (Count)
Synthetic Cars [14]	**Car Model**	6	"a photo of a [car model]"	Volvo S60, BMW X5 M, ... (569)
	Car Color	18	"a [color name] car"	orange, black, ... (18)
	Background Color	18	"a car in front of a [color] background"	orange, black, ... (18)
Cars196 [15]	**Car Model**	98	"a photo of a [car model]"	Volvo S60, BMW X5 M, ... (569)
	Manufacturer	35	"a photo of a car produced by [manufacturer]"	Tesla, BMW, ... (46)
	Car Type	7	"a photo of a [car type]"	convertible, SUV, ... (7)
CUB200 [29]	**Bird Species**	100	"a photo of a [bird species]"	Black footed Albatross, Rusty Blackbird, ... (100)
DeepFashion [16]	**Clothing Category**	50	"a photo of a person wearing a [clothing category]"	anorak, turtleneck, ... (50)
	Texture	7	"a photo of a person wearing clothes with a [texture type] texture"	floral, striped, ... (7)
	Fabric	6	"a photo of a person wearing clothes made out of [fabric type]"	cotton, leather, ... (6)
	Fit	3	"a photo of a person wearing clothes with a [fit type] fit"	tight, loose, conventional (3)
Movie Posters [3]	**Genre**	25	"a poster of a [genre] movie"	Comedy, Action, ... (25)
	Production Country	69	"a poster of a movie produced in [country]"	USA, India, ... (69)

Nonlinear Autoencoder (AE) While PCA and LAE are linear models, we also test a more powerful nonlinear Autoencoder, which consists of a two-layer encoder and decoder with 512 hidden units and leaky ReLU activation functions [17]. We use the same loss function and hyperparameters as for LAE, but add a weight decay of 10^{-2} to alleviate overfitting on the few text prompts.

Oracle InDiReCT uses only text prompts to optimize the transformation matrix U that maps CLIP embeddings to a more specialized, lower-dimensional unit hypersphere. To estimate how well InDiReCT could theoretically perform, we employ an Oracle that optimizes U directly on *test images and their labels*. For this, we use the common DML loss function Normalized Softmax Loss [33]. We first compute unit-length image embeddings v_i' as in Equations (5) and (6) and then optimize the transformation matrix U to minimize the loss function $L_{Oracle} = \frac{1}{m}\sum_{i=1}^{m} -\log\left(\frac{\exp(v_i' c_{l_i}^\top)}{\sum_j \exp(v_i' c_j^\top))}\right)$, where m is the number of test images and $c_{l_i} \in \mathbb{R}^{1\times r'}$ with $\|c_{l_i}\| = 1$ is the prototype vector of the class for the label of the ith image l_i, which is optimized jointly with U using Adam (learning rate 0.01, early stopping with patience 100).

Note that in Language-Guided Zero-Shot Deep Metric Learning, neither images nor their labels are available for training. We use this baseline method in order to provide a *very optimistic* estimate of what performance InDiReCT could achieve given perfect information. The Normalized Softmax Loss is a classification-based training objective, so image embeddings are processed independently. Thus, the loss does not optimize for the best nearest neighbor performance, i.e. Precision@1. To compare the Oracle baseline to other models, we thus primarily use MAP@R.

Low (high) Oracle performance can be used to identify similarity notions that cannot (can) be reliably represented using InDiReCT since they are not captured (are captured) in the CLIP embeddings. If InDiReCT's performs substantially worse than the Oracle, it means that the text prompts were not capable of capturing the desired similarity notion.

5. Results

We report the mean and standard deviation of the evaluation metrics over five runs in Table 2. The CLIP baseline typically achieves substantially better results than the random baseline. Since the embeddings stay the same in each run, its performance does have a standard deviation of zero and is omitted for brevity. Despite the fact that the CLIP baseline uses four times larger embedding vectors, InDiReCT almost always performs better than CLIP and achieves the best performance in most datasets and similarity notions. Depending on the dataset and similarity notion, InDiReCT can improve CLIP's MAP@R score by up to 14 percentage points. Switching the learned matrix to a random transformation matrix in InDiReCT usually performs worse than CLIP. As described in Section 4.2, PCA is only applicable to two datasets and similarity notions. There, InDiReCT and PCA perform similarly. Training a Linear Autoencoder (LAE) on the text embeddings usually improves the CLIP baseline, but does not achieve better performance than InDiReCT. Applying a more complex nonlinear Autoencoder performs oftentimes worse than the CLIP baseline and also shows substantially larger standard deviations, which might be due to the model not handling the few datapoints well. These results show that choosing a suitable dimensionality reduction technique can improve performance and opens up new research directions. In general, InDiReCT learns a useful embedding function by using text prompts that describe different aspects of the desired similarity notion.

The Oracle baseline is optimized directly on the image dataset and their labels. Despite all this, InDiReCT matches or exceeds the Prec@1 performance of the Oracle baseline for Cars196, CUB200, and the "Genre" similarity notion for the Movie Posters dataset. As discussed in Section 4.2, this might be due to the classification-based nature of the Normalized Softmax Loss. For MAP@R, the Oracle is the

Table 2. Results for our experiments. All values are given in percent, best in bold.

			Random	CLIP (512-dim.)	InDiReCT	Rand. trans.	PCA	LAE	AE	Oracle
Synthetic Cars	**Car Model**	**MAP@R**	3.3 ± 0.1	43.5	**57.4 ± 0.2**	39.1 ± 1.6	56.2 ± 0.1	52.5 ± 0.5	39.5 ± 4.4	100 ± 0.0
		Prec@1	17.5 ± 0.9	95.4	96.4 ± 0.0	93.4 ± 0.5	**96.6 ± 0.1**	95.9 ± 0.5	88.7 ± 3.6	100 ± 0.0
	Car Color	**MAP@R**	5.0 ± 0.1	6.2	**9.1 ± 0.1**	6.1 ± 0.1	—	7.3 ± 0.2	8.6 ± 0.4	57.9 ± 0.9
		Prec@1	17.5 ± 0.8	27.6	**31.4 ± 0.5**	26.3 ± 1.3	—	29.4 ± 0.9	30.2 ± 1.3	79.3 ± 0.8
	Background Color	**MAP@R**	5.4 ± 0.0	6.2	**7.1 ± 0.0**	6.1 ± 0.2	—	6.3 ± 0.2	6.1 ± 0.2	74.0 ± 0.9
		Prec@1	19.4 ± 1.1	27.0	**28.3 ± 0.3**	26.6 ± 1.1	—	**28.3 ± 0.7**	21.6 ± 1.3	88.0 ± 0.4
Cars196	**Car Model**	**MAP@R**	0.1 ± 0.0	23.5	37.4 ± 0.0	19.2 ± 0.3	**37.5 ± 0.1**	33.2 ± 0.2	20.0 ± 5.8	41.8 ± 0.0
		Prec@1	1.1 ± 0.1	78.0	**84.4 ± 0.1**	72.9 ± 0.5	84.2 ± 0.1	82.4 ± 0.2	63.8 ± 8.1	*76.6 ± 0.1*
	Manufacturer	**MAP@R**	0.5 ± 0.0	24.4	**33.6 ± 0.1**	21.2 ± 0.4	—	24.2 ± 0.4	18.0 ± 2.2	51.4 ± 0.0
		Prec@1	5.4 ± 0.3	89.0	**90.5 ± 0.1**	84.7 ± 0.8	—	85.5 ± 0.3	63.1 ± 3.9	*84.0 ± 0.1*
	Car Type	**MAP@R**	3.5 ± 0.0	25.1	**36.1 ± 0.3**	22.1 ± 0.8	—	27.7 ± 0.6	24.4 ± 1.6	73.8 ± 0.0
		Prec@1	17.3 ± 0.4	**91.1**	90.7 ± 0.2	88.3 ± 0.5	—	89.1 ± 0.4	63.2 ± 3.1	*89.1 ± 0.0*
CUB200	**Bird Species**	**MAP@R**	0.1 ± 0.0	18.0	**26.5 ± 0.0**	15.2 ± 0.3	—	18.8 ± 0.2	15.1 ± 1.9	34.1 ± 0.0
		Prec@1	1.2 ± 0.1	58.2	**65.3 ± 0.1**	52.6 ± 0.3	—	58.1 ± 0.5	44.4 ± 3.6	*65.3 ± 0.2*
DeepFashion	**Clothing Category**	**MAP@R**	2.3 ± 0.0	12.5	**18.7 ± 0.1**	11.3 ± 0.4	—	13.3 ± 0.3	16.9 ± 1.8	32.2 ± 0.1
		Prec@1	11.1 ± 0.4	45.2	**50.9 ± 0.2**	43.0 ± 0.6	—	45.5 ± 0.5	44.5 ± 2.4	55.8 ± 0.6
	Texture	**MAP@R**	11.8 ± 0.0	18.7	**33.0 ± 0.4**	11.2 ± 0.4	—	22.2 ± 0.5	16.3 ± 0.7	66.1 ± 0.1
		Prec@1	29.6 ± 0.7	60.2	**66.8 ± 0.3**	43.3 ± 0.5	—	61.2 ± 0.7	43.8 ± 1.7	80.6 ± 0.3
	Fabric	**MAP@R**	32.4 ± 0.0	34.0	**37.7 ± 0.2**	10.8 ± 0.3	—	35.6 ± 0.3	17.2 ± 0.6	64.2 ± 0.3
		Prec@1	49.4 ± 0.6	64.5	**66.1 ± 0.6**	42.6 ± 0.7	—	65.1 ± 0.6	44.7 ± 1.9	77.8 ± 0.4
	Fit	**MAP@R**	51.8 ± 0.0	53.3	**53.9 ± 0.4**	11.1 ± 1.0	—	53.4 ± 0.3	16.1 ± 1.8	82.0 ± 0.1
		Prec@1	66.6 ± 0.6	**77.1**	76.5 ± 0.4	43.1 ± 0.5	—	76.7 ± 0.7	42.9 ± 1.9	87.8 ± 0.6
Movie Posters	**Genre**	**MAP@R**	4.1 ± 0.0	11.4	**14.9 ± 0.0**	9.1 ± 0.3	—	8.4 ± 0.1	9.8 ± 2.4	19.6 ± 0.1
		Prec@1	17.5 ± 0.4	41.8	**44.0 ± 0.2**	38.1 ± 0.7	—	36.6 ± 0.4	33.3 ± 3.0	*43.2 ± 0.7*
	Production Country	**MAP@R**	44.6 ± 0.0	49.3	**51.3 ± 0.1**	48.9 ± 0.4	—	47.7 ± 0.2	49.4 ± 0.7	58.1 ± 0.0
		Prec@1	59.2 ± 0.5	69.3	**69.8 ± 0.3**	67.9 ± 0.7	—	68.1 ± 0.3	64.9 ± 0.7	71.8 ± 0.3

best model for all datasets and similarity notions.

Even though the comparison is not fair, we contrast InDiReCT's performance with state of the art models from the literature that train on a large labeled training dataset regarding the desired similarity notion. Note that only Cars196's "Car Model" and CUB200's "Bird Species" similarity notions have been used in the literature in a DML setting, so we only compare to them. Jun et al. [10] achieve Prec@1 of 94.8 and 79.2 for Cars196 and CUB200, respectively [27], which outperform InDiReCT by ten to fourteen percentage points. However, the trained models output 1536-dimensional vectors, more than ten times the embedding dimensions we use in our experiments. For embeddings of dimensions 128, Jun et al. achieve 90.1 (Cars196) and 67.6 (CUB200) Prec@1, which is only approximately six and two percentage points better than InDiReCT. These results show that despite not using any training images, InDiReCT can show strong performance even compared to fully supervised methods.

6. Analysis

What does InDiReCT attend to in the input? We want to visualize the image regions that are used by InDiReCT to output a certain embedding. Due to the positive experimental results, we assume that, for a given similarity notion, InDiReCT attends to subjectively more useful regions than CLIP. We thus compute saliency maps using the method introduced by Kobs et al. [14] and subtract InDiReCT's saliency maps from CLIP's saliency maps to qualitatively showcase the difference between both methods.

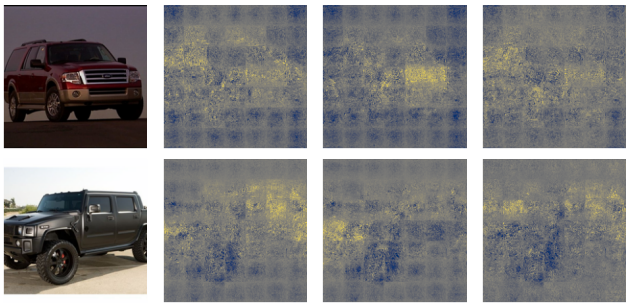

Figure 2. Example images from the Cars196 dataset and the saliency map differences between each similarity notion and CLIP. InDiReCT focuses more on yellow regions, CLIP more on blue regions. The patch patterns in the images are due to the patch creation of CLIP's Vision Transformer [4]. More examples in the appendix.

We choose Cars196 and its similarity notions and hypothesize that InDiReCT pays more attention to regions that represent the desired similarity notion than CLIP. In order to increase the chance of obtaining visible differences in the saliency maps, we reduce the number of embedding dimensions for InDiReCT to two, thus only extracting the most important features to embed the given images. Figure 2 shows two example images (more in the appendix). Yellow areas indicate image regions InDiReCT pays more attention

to than CLIP, while CLIP focuses more on blue regions. Grey areas show similarly strong saliency.

Compared to CLIP, InDiReCT focuses more on the area of the car when using the "Car Model" similarity notion, which is useful for the task. Interestingly, for "Manufacturer", InDiReCT mostly uses the front of the car, where the manufacturer's logo is usually found. Additionally, the design of the radiator grill and headlights is often relatively unique to manufacturers. For the "Car Type" similarity notion, InDiReCT focuses more on the back of the car, as car types such as "convertible", "van", or "sedan" differ mainly in terms of trunk and roof design.

Do other embedding sizes perform differently? While our experiments set the embedding size arbitrarily to 128, we now measure the performance on the Cars196 dataset with varying target embedding dimensions $r' \in \{2, 4, 8, \dots, 256, 512\}$. We plot the MAP@R mean and standard deviation over five runs for all methods and all similarity notions in Figure 3. CLIP with its fixed 512 dimensions is plotted as a reference line.

InDiReCT matches or exceeds CLIP's performance when using at least 16 embedding dimensions and peaks at 64 dimensions for all three similarity notions. The learned transformation presumably selects, combines, and weights CLIP's embedding dimensions such that InDiReCT even outperforms CLIP for 512 dimensions.

Do larger CLIP models improve performance? For our experiments, we use the CLIP model "ViT-B/32" [26], i.e. a Vision Transformer [4] with 12 layers and input patches of size 32×32 pixels. We now test larger CLIP models as feature extractors in InDiReCT with CLIP's "ViT-B/16" and "ViT-L/14" versions, which change the input patches to 16×16 and 14×14 pixels, respectively, while "ViT-L/14" also doubles the transformer layers. Besides other parameters, "ViT-L/14" also increases CLIP's outputs from 512- to 768-dimensional vectors.

We test all three ViT sizes to see if larger CLIP versions lead to better performance [26]. The "Synthetic Cars" dataset with its similarity notions is used, since the performance of InDiReCT is quite good for "Car Model", but bad for "Car Color" and "Background Color", compared to the Oracle baseline. With this analysis, we can investigate whether larger models can improve performance for these similarity notions. We use 128 embedding dimensions.

Figure 4 shows that the performance of the Oracle baseline increases with larger models, which means that the model extracts more useful features that could potentially be picked up by InDiReCT. For the "Car Model" similarity notion, this also translates to better performance of InDiReCT and CLIP in general. On the other two similarity notions, however, we cannot find any performance improvements. Since the Oracle baseline improves, we can conclude that the text prompts used to train InDiReCT lead to a focus on suboptimal features for these similarity notions. Other text prompts might increase performance.

Do more text prompts improve performance? Our final analysis takes a closer look at how the performance of InDiReCT changes if we use different numbers of prompts for our experiments. We use the Cars196 dataset and focus on the "Car Model" similarity notion. Originally, we use 569 different car model names from an online car dealer as a basis for the text prompts ("a photo of a [car model name]"). We now sample differently sized sets from these car model names and run our experiment five times with different samples. Figure 5 shows the means and standard deviations for sizes $\{10, 20, \dots, 150\}$. The performance increases with larger sample sizes and converges at around 90 prompts to the performance we observe in our main experiments. This behavior is expected, since the learned transformation is able to better capture the important dimensions in the text embeddings when more prompts are used. For fewer prompts, InDiReCT can almost perfectly reconstruct the text embeddings, thus is not forced to select the important dimensions. Figure 5 also shows that with larger prompt sets, the standard deviation of performance tends to decrease. Overall, we can observe that more (useful) text prompts should stabilize and improve performance for InDiReCT.

7. Discussion

Using natural language, the proposed LanZ-DML setting offers a simple interface for adapting item retrieval systems to the desired similarity notion. This adaption is not achievable using raw CLIP embeddings or other self-/unsupervised methods. For InDiReCT, it is not necessary to collect and annotate example images, which is time-consuming and tedious. Expressing the desired similarity notion using text prompts is certainly simpler, but limits its application to similarity notions with categorical aspects. However, this is a limitation that also holds for popular proxy-based DML loss functions such as Normalized Softmax Loss [33] or ProxyNCA [21], i.e., loss functions that use class prototype vectors. It should also be noted that the quality of text prompts might vary significantly. In our experiments, we comply with the zero-shot setting by choosing plausible prompt templates without validating them on the data. Overall, we achieve good performance across datasets and similarity notions. However, as already shown for prompt engineering [26], there might be prompts that work substantially better. Often, exploiting the peculiarities of the dataset CLIP has been trained on helps. For example, instead of using single words as text prompts, short sentences usually work better [26]. Therefore, it is recommended to test different text prompts when applied in real-world scenarios.

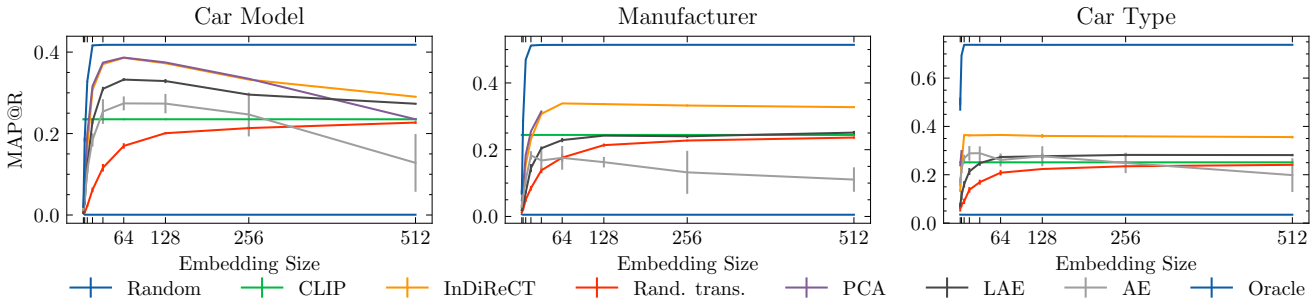

Figure 3. On Cars196, InDiReCT outperforms other zero-shot models for embedding sizes 16 and up, while it peaks at 64 dimensions.

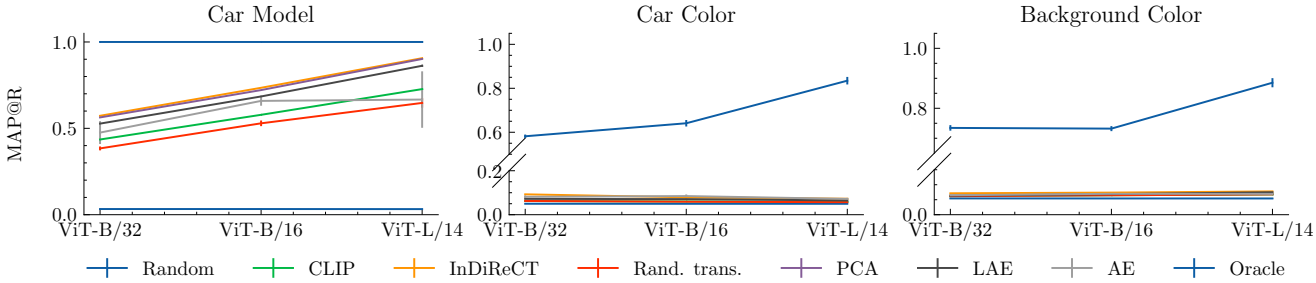

Figure 4. Larger CLIP models improve performance for the "Car Model" but not for color similarity notions on Synthetic Cars.

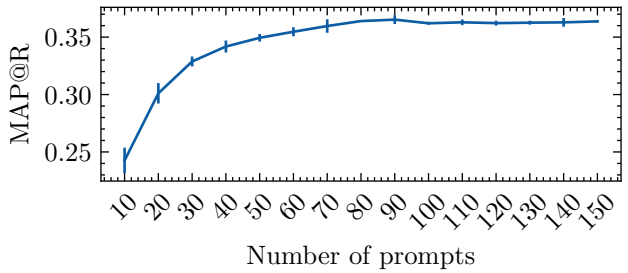

Figure 5. Performance of InDiReCT for different number of training prompts. We sample different car model names for each run.

Also, tuning the number of embedding dimensions is not straightforward without validation data, leading to suboptimal performance when using 128- instead of 64-dimensional vectors for the Cars196 dataset, as shown in our analysis.

Since we use CLIP as a fixed feature extractor, we need to rely on the usefulness of its embeddings. If CLIP does not extract properties from images and texts related to a desired similarity notion, InDiReCT cannot show its full potential. We have shown that InDiReCT mostly outperforms CLIP, so the text prompts help to focus on the desired similarity notion. Given the Oracle results, however, some datasets and similarity notions (e.g. Synthetic Cars' color notions) could potentially work better. In some cases, larger CLIP models can improve the performance as shown in our analysis.

Since we use pretrained CLIP embeddings and only a handful of text prompts, training the dimensionality reduction is fast. It also allows us to precompute CLIP embeddings for a whole image database and adaptively transform them with a trained dimensionality reduction. The disadvantage of this is that, for each search, the transformation matrix must be applied to all vectors in the image collection. Potentially, existing vector search databases [30, 9] can efficiently incorporate the transformation to retrieve relevant images.

8. Conclusion

In this paper, we have introduced Language-Guided Zero-Shot Deep Metric Learning (LanZ-DML), a setting where no training data and labels but only texts are allowed to guide a Deep Metric Learning model for a given similarity notion. Our proposed model InDiReCT is based on fixed CLIP embeddings of text prompts describing the varying aspects of a given similarity notion. We have shown that InDiReCT outperforms strong baselines and approaches fully supervised methods. Our analyses show that InDiReCT focuses on image regions that are subjectively important for the desired similarity notion. We have also investigated the influence of different hyperparameters on the model performance.

Due to its simple design and fast training, InDiReCT can be useful for users to customize the similarity notion of item retrieval systems. The need to define multiple prompts based on the changing aspects of a similarity notion could be facilitated, e.g. by directly learning the transformation from sentences such as "Two car images are similar if both cars are the same model". Automatic selection of hyperparameters and developing methods for LanZ-DML on other modalities, e.g. audio or texts, are also interesting research directions.

References

[1] Alberto Baldrati, Marco Bertini, Tiberio Uricchio, and Alberto Del Bimbo. Conditioned image retrieval for fashion using contrastive learning and clip-based features. In *ACM Multimedia Asia*, pages 1–5. 2021.

[2] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation learning. In *European conference on computer vision*, pages 104–120. Springer, 2020.

[3] Wei-Ta Chu and Hung-Jui Guo. Movie genre classification based on poster images with deep neural networks. In *Proceedings of the Workshop on Multimodal Understanding of Social, Affective and Subjective Attributes*, pages 39–45, 2017.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[5] Marsaglia George. Choosing a point from the surface of a sphere. *Ann. Math. Statist*, 43:645–646, 1972.

[6] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Discriminative deep metric learning for face verification in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1875–1882, 2014.

[7] Zhanxuan Hu, Danyang Wu, Feiping Nie, and Rong Wang. Generalization bottleneck in deep metric learning. *Information Sciences*, 581:249–261, 2021.

[8] Mengdi Huai, Hongfei Xue, Chenglin Miao, Liuyi Yao, Lu Su, Changyou Chen, and Aidong Zhang. Deep metric learning: The generalization analysis and an adaptive algorithm. In *IJCAI*, pages 2535–2541, 2019.

[9] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[10] HeeJae Jun, Byungsoo Ko, Youngjoon Kim, Insik Kim, and Jongtack Kim. Combination of multiple global descriptors for image retrieval. *arXiv preprint arXiv:1903.10663*, 2019.

[11] Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[12] Lyman M Kells. *Plane and Spherical Trigonometry with Tables by Lyman M. Kells, Willis F. Kern, James R. Bland*. US Armed Forces Institute, 1940.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Konstantin Kobs, Michael Steininger, Andrzej Dulny, and Andreas Hotho. Do different deep metric learning losses lead to similar learned features? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10644–10654, 2021.

[15] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[16] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[17] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, volume 30, page 3. Citeseer, 2013.

[18] Danny Merkx, Stefan L Frank, and Mirjam Ernestus. Language learning using speech to image retrieval. *arXiv preprint arXiv:1909.03795*, 2019.

[19] Timo Milbich, Karsten Roth, Biagio Brattoli, and Björn Ommer. Sharing matters for generalization in deep metric learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):416–427, 2020.

[20] Timo Milbich, Karsten Roth, Samarth Sinha, Ludwig Schmidt, Marzyeh Ghassemi, and Bjorn Ommer. Characterizing generalization under out-of-distribution shifts in deep metric learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[21] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 360–368, 2017.

[22] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, pages 681–699. Springer, 2020.

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] Elad Plaut. From principal subspaces to principal components with linear autoencoders. *arXiv preprint arXiv:1804.10253*, 2018.

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[27] Steffen Rendle. Evaluation metrics for item recommendation under sampling. *arXiv preprint arXiv:1912.02263*, 2019.

[28] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Björn Ommer, and Joseph Paul Cohen. Revisiting train-

ing strategies and generalization performance in deep metric learning, 2020.

[29] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[30] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627, 2021.

[31] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[32] Xinyi Xu, Huanhuan Cao, Yanhua Yang, Erkun Yang, and Cheng Deng. Zero-shot metric learning. In *IJCAI*, pages 3996–4002, 2019.

[33] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning. *arXiv preprint arXiv:1811.12649*, 2018.