# Image Retrieval with Self-Supervised Divergence Minimization and Cross-Attention Classification

**Vivek Trivedy , Longin Jan Latecki**
Department of Computer and Information Sciences, Temple University, Philadelphia
{vivek.trivedy, latecki}@temple.edu

## Abstract

Common approaches for image retrieval include contrastive methods and specialized loss functions such as ranking losses and entropy regularizers. We present **DMCAC** (Divergence Minimization with Cross-Attention Classification) which offers a new perspective on this training paradigm. We use self-supervision with a novel divergence loss framework alongside a simple data flow adjustment that minimizes the distributional divergence over a database directly during training. We show that jointly learning the query representation over a database is a competitive and often improved alternative to contrastive and other methods for image retrieval. We evaluate our method across several model configurations and four datasets, achieving state-of-the-art performance in multiple settings. We also conduct a thorough set of ablations that show the robustness of our method across full vs. approximate retrieval and different hyperparameter configurations.

## 1 Introduction

Representing similarity between entities is critical to many tasks such as content-based image retrieval, cross-modal information retrieval, face recognition, and person re-identification. A key driver behind these methods is metric learning where the task is to learn embeddings such that their distances preserve a notion of semantic similarity. There are several variations on metric learning losses, which include ideas such as sampling [Yu *et al.*, 2019; Robinson *et al.*, 2021] and different schemes for grouping examples together such as pair-based [Khosla *et al.*, 2021; Hoffer and Ailon, 2018] and proxy-based [Kim *et al.*, 2020a; Movshovitz-Attias *et al.*, 2017b]. The overall goal behind metric learning methods is to train an embedding model to pull similar objects together in the embedding space while (optionally) pushing dissimilar objects apart. Similarity is often given by a class label where all examples in a dataset of class $c$ are pulled together and examples of other classes are pushed apart. Other options for determining similarity include self-supervised learning [Caron *et al.*, 2021b; Caron *et al.*, 2019; Caron *et al.*, 2021a] where an input image is transformed into a pair or set of views via data augmentation. Here, all views of the same image are ”similar.” In the metric learning for image retrieval setting, the encoder often takes one of two forms: a convolutional neural network (CNN) [He *et al.*, 2015; Szegedy *et al.*, 2014] or more recently a Vision Transformer [Dosovitskiy *et al.*, 2021; Touvron *et al.*, 2021]. The goal of the encoder is to transform each input image into a learned embedding optimized for the downstream task of retrieval from an external set, usually called a database. An important point to note is that the training setting is decoupled from the retrieval setting in that there is no database present during training so queries are not conditioned over a database. Instead there is an auxiliary task such as a metric learning objective which is then used for retrieval during evaluation. Here we propose an approach that instead jointly learns query and database representations.

We offer a novel framework for training image retrieval systems which we call DMCAC (Divergence Minimization with Cross-Attention Retrieval). It offers a few novel insights. Firstly, we use a query set and database set directly during training. We generate and embed several views of an input query via data augmentation as is common in the self-supervised regime. But rather than minimizing the distance between each query view, we first compute a similarity distribution between the query views and a separate database set prodcing a similarity distribution for each view over the database. Our method is motivated by *distribution matching* which we empirically find to be stronger than *point-wise* objectives like triplet loss. For our loss, we minimize the divergence in this similarity distribution between views so that our loss is conditioned from a query set over a database set, rather than over a query set alone. This formulation closely resembles the retrieval test setting where a query set is used to retrieve similar sets of items from the database.

Importantly, we note that the self-supervised method is prone to collapse [Caron *et al.*, 2021a] where the encoder ”cheats” by simply embedding all views to the same point because there is no grounding signal of ”correctness” such as a class label. We overcome this issue by using the retrieved database embeddings to also directly predict the query class label via cross-attention. In this way, the retrieved examples must learn some notion of semantic meaning about the query. Our key contributions are:

- We present a novel retrieval framework that applies self-supervised learning to directly learn a joint representation between a query set and a database set. This approach offers a competitive and often improved alternative to common metric learning retrieval frameworks.

- We introduce a novel loss for retrieval. We minimize divergence between query views via a Frobenius norm and classification error via cross-attention with cross-entropy using query images conditioned over a database set.
- We investigate the tradeoffs between end-to-end differentiability and efficient approximate nearest neighbor (ANN) search alongside a thorough ablation of hyperparameters. We show that our method is competitive across both settings.

## 2 Related Works

### 2.1 Transformers and Vision Transformers

Transformers [Vaswani *et al.*, 2017] revolutionized natural language processing by using self-attention and blocks of fully-connected layers. Models such as BERT [Devlin *et al.*, 2019] were highly performant across a broad spectrum of NLP tasks such as text classification and language translation. Transformers excel in capturing local dependencies and using them to produce global representations which made them an attractive candidate for computer vision. The key breakthrough came with Vision Transformers (ViTs) [Dosovitskiy *et al.*, 2021] which had the insight that local tokens in an image are simply patches which can then be fed to the Transformer architecture. ViTs match or surpass convolutional models across tasks such as image classification [Caron *et al.*, 2021b; Chen *et al.*, 2021] and semantic segmentation [Strudel *et al.*, 2021]. Because ViTs lack the traditional inductive biases of CNNs, methods arose to efficiently train ViTs [Steiner *et al.*, 2022]. These methods include using data augmentation [Cubuk *et al.*, 2019b], knowledge distillation [Touvron *et al.*, 2021], pretraining with larger datasets [Koppula *et al.*, 2022], and specialized architectural variants such as Swin Transformer [Liu *et al.*, 2021] which reintroduce vision specific biases.

### 2.2 Metric Learning

There are many approaches to Metric Learning, but here we give an overview of pairwise and classification approaches. Contrastive loss [Carreira-Perpiñán and Hinton, 2005; Hadsell *et al.*, 2006] and triplet loss [Hoffer and Ailon, 2018] are examples of pairwise losses where the goal is to bring similar examples together while pushing dissimilar examples away in embedding space by generating groups of examples. Other variations such as Supervised Contrastive Loss [Khosla *et al.*, 2021] introduce class information that helps create positive and negative pairs conditioned on a semantic label. These methods can be further improved by employing techniques such as Hard Negative Sampling [Robinson *et al.*, 2021] and other sampling methods [Yu *et al.*, 2019] to ensure that the network avoids collapse from an oversaturated signal of easy examples during optimization. Methods such as [El-Nouby *et al.*, 2021a; Song *et al.*, 2023] show the effectiveness of combining contrastive methods with Vision Transformers for retrieval.

Classification based approaches represent categories with a single or multiple representative points [Movshovitz-Attias *et al.*, 2017a; Teh *et al.*, 2020a; Boudiaf *et al.*, 2021]. In these methods, a new example is compared against the representative points, and a class is determined based on similarity to each class representative [Zhai and Wu, 2019a; Trivedy and Latecki, 2023]. All of these methods show that direct classification is a viable and performant technique for retrieval. We draw inspiration from these approaches in designing our novel classification approach which uses cross-attention over a database set.

### 2.3 Self-Supervised Learning

Self-supervised learning (SSL) constructs its learning signal from the data itself where a model learns to predict a portion of the input data from other parts, compelling it to learn robust, high-level features. The key benefit here is that this requires no labeled data and is thus a good approach for large-scale pretraining [He *et al.*, 2021; Baevski *et al.*, 2022]. Common SSL approaches include solving pretext tasks, masking, and learning invariances to data augmentations. SSL pretext tasks include [Doersch *et al.*, 2016] which predicts the relative position of image patches, and [Noroozi and Favaro, 2017a; Noroozi and Favaro, 2017b] which use jigsaw puzzles where a model learns to recognize the permutation of shuffled image patches. Masking methods such as [He *et al.*, 2021] are trained to reconstruct masked portions of an image. In this work, we focus on augmentation based SSL methods such as SimCLR [Chen *et al.*, 2020] which generate multiple views of an image via data augmentation and minimize the embedding distance between them. These methods are often paired with contrastive learning where positive pairs are two augmented views of a single image and negative pairs are views from two different images.

Self-supervised learning has benefited from work on selecting performant data augmentation strategies [Cubuk *et al.*, 2019b; Cubuk *et al.*, 2019a]. Reducing the embedding distance between variations of the same image (via cropping, color jittering, etc.), teaches a model to be invariant to these changes while hopefully preserving the semantic meaning of the image which leads to more robust and generalizable features [Park *et al.*, 2023].

## 3 Method

In this section, we outline the details of our approach. In particular, we describe the building blocks for training our model, define our dataset contruction, highlight key design choices in our forward pass, and discuss the retrieval mechanism and loss function.

### 3.1 Dataset Construction

To motivate our approach we share the rationale and process for constructing the training and testing sets, following common practices of usual data splits. Because we want to

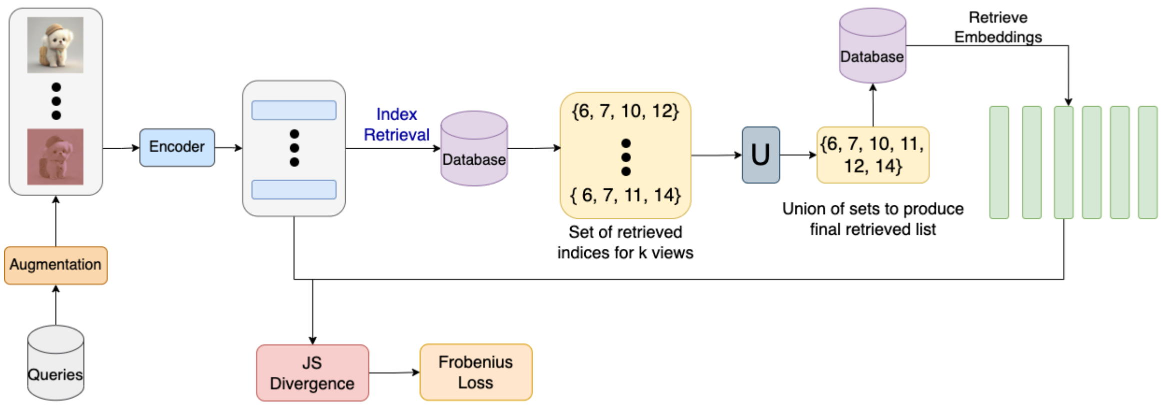

Figure 1: The data flow of our method when computing the Frobenius loss. In the forward pass, embeddings of different views of the same query retrieve $k$ nearest neighbor indices from the database. After computing the union of the indices over the query views, their associated embeddings are retrieved and used to compute the Frobenius loss.

directly train our model on the task of retrieval, we construct four sets which we describe here. Please see Section 4.1 for detailed information on each dataset.

- Training Query Set: $\mathcal{D}_Q$
- Training Database Set: $\mathcal{D}_D$
- Evaluation Query Set: $\mathcal{D}'_Q$
- Evaluation Database Set: $\mathcal{D}'_D$

Note that the training and evaluation sets have a disjoint set of classes. In this way, during training, we learn general representations that can be extended easily to new classes which is a common practice in retrieval evaluation.

## 3.2 Revisiting Vision Transformers

In our approach, we leverage Vision Transformers (ViTs). Here we provide an overview of their building blocks. A ViT operates over sequences like the original transformer [Vaswani *et al.*, 2017] architecture. The input sequence can be described by

$$X = [\mathbf{x}_{cls}; \mathbf{x}_1; \ldots; \mathbf{x}_M] \in \mathbb{R}^{(M+1)\times F} \tag{1}$$

Here each $\mathbf{x}_i$ represents a patch token with dimension $F$ of which there are $M$, and the global representation for the image is captured by $\mathbf{x}_{cls}$. The most common approach for splitting an image into a set of patches is to create non-overlapping patches of shape $P \times P$ to produce a sequence of patch tensors of shape $P \times P \times C$ where $C$ is the number of channels. Each patch is flattened to a vector of size $P^2C$ and embedded to dimension $F$ via a shared linear layer. This is equivalent to applying a square convolutional filter with kernel size $P$ and stride $P$.

After generating the sequence of patches, learnable positional encodings are added so that the model is aware of spatial information instead of being a pure set-to-set model. The ViT then performs $K$ blocks of computation consisting of multi-head self-attention (MHSA), layer normalization [Ba *et al.*, 2016], feed-forward layers (FFN), and a skip connection. The input sequence length and dimensionality are preserved in each block. The output we use is an embedding $z_{cls}$ which captures the global representation of the input image.

## 3.3 Forward Pass

Here we present the data flow through our model and our augmentation setup. We define our training objective, namely to learn an updated encoder function $\phi$ via a base encoder, training query set, and training database set.

$$\phi_{new} = \mathcal{F}(\phi_{new}, \mathcal{D}_Q, \mathcal{D}_D), \tag{2}$$

where initially $\phi_{new} = \phi_{base}$. In this case, our base encoder model, $\phi_{base}$ is a flavor of ViT that comes initialized with pretrained weights. We first embed each example in our training database set $\mathcal{D}_D \in \mathbb{R}^{D\times C\times H\times W}$ via our base encoder, where $D$ is the number of the training database images.

$$Z_D = \phi_{new}(\mathcal{D}_D) \in \mathbb{R}^{D\times F} \tag{3}$$

Here we produce $Z_D$ from our database where each of the $D$ rows is an image embedding with output dimension $F$. This is achieved by outputting the *CLS* token for each image from the penultimate layer of the ViT and concatenating each of the tokens row-wise.

$$Z_D = [cls_1, \ldots, cls_D] \tag{4}$$

After constructing the database embeddings, we begin training the base encoder. We present our approach for a single query, but the method extends naturally to the minibatch setting. From our training query set, $\mathcal{D}_Q$, we select an image and perform data augmentation to generate $A$ views of the image (one of them is the original image) and produce $X_A \in \mathbb{R}^{A\times C\times H\times W}$, where $H$ and $W$ are image dimensions and $C$ is the number of channels. We then embed each image view with the encoder

$$Z_A = \phi_{new}(X_A) \in \mathbb{R}^{A\times F}, \tag{5}$$

where $Z_A$ is the concatenated set of $CLS$ tokens for each of the views in the minibatch. As is common in other methods [Caron *et al.*, 2021a; El-Nouby *et al.*, 2021a], we project the output embedding for each view onto the unit ball for each view $\mathbf{x}_i$ via $\phi_{new}(\mathbf{x}_i)/\|\phi_{new}(\mathbf{x}_i)\|_2$. We compute the same projection for the database set so a dot product similarity computed between the embedded query views and database set is equivalent to cosine similarity.

## 3.4 Training Retrieval Mechanism

We begin with retrieval over all database image embeddings, $Z_D$, using our query embeddings, $Z_A$. Hence, the full retrieval computes the cosine similarity exhaustively over all database embeddings $Z_D$

**Full Retrieval Setting.** We compute a similarity matrix $P$ as a product between the embeddings of the query views, $\mathbf{z}_A$, and the transpose of $Z_D$:

$$P = Z_A \cdot (Z_D)^{\mathbf{T}} \in \mathbb{R}^{A \times D} \tag{6}$$

$$P' = \sigma(P) = \frac{e^{P_{ij}}}{\sum_{h=1}^{T} e^{P_{ih}}} \quad for\ i = 1, 2, \ldots, D, \tag{7}$$

where $\sigma$ is the softmax operator, so each row of $P'$ can be interpreted as a probability distribution. We use $P'$ to compute our loss as described in Section 3.5.

Note that in this approach we receive a full gradient signal from the database to the encoder. This setting acts as an implicit negative sampling where the encoder receives signals from both similar and dissimilar embeddings.

However, this simple setting only makes sense if all embeddings of the database images fit into GPU memory, which is an unrealistic assumption for larger databases. To address this limitation, we consider an approximate retrieval setting, where first retrieve embeddings of top-k images, and use them to construct the loss function defined in Section 3.5. These two methods differ in that approximate nearest neighbor retrieval uses FAISS [Johnson *et al.*, 2019] with cosine similarity while full retrieval computes the cosine similarity exhaustively over all database embeddings $Z_D$.

**Approximate Retrieval Setting.** Let $\mathbf{z} \in \mathbb{R}^{1 \times F}$ denotes a deep embedding of query $q$, and let $Z_A \in \mathbb{R}^{A \times F}$ denotes embeddings of the query views, where row $i$ is an embedding of view $q_i$ with $q_1 = q$. For each view $q_i$ of $q$, we retrieve database indices of its $k$ nearest neighbors, denoted as $S_i$. From here, we use set operations to generate a combined set of retrieved indices across all views. The goal is to find the full set of indices retrieved by all of the views together:

$$S^{union} = \bigcup_{j=1}^{A} S_j. \tag{8}$$

Now $S^{union}$ is a set containing all indices of the nearest neighbors across all $A$ query views. We then retrieve the associated embeddings from the database to produce $Z_S^{union} \in \mathbb{R}^{T \times F}$, where $T$ is the cardinality $|S^{union}|$. Finally, we compute a similarity matrix $P$ as a product between the embeddings of the query views, $\mathbf{z}_A$, and the transpose of $Z_S^{union}$:

$$P = Z_A \cdot (Z_S^{union})^{\mathbf{T}} \in \mathbb{R}^{A \times T} \tag{9}$$

$$P' = \sigma(P) = \frac{e^{P_{ij}}}{\sum_{h=1}^{T} e^{P_{ih}}} \quad for\ i = 1, 2, \ldots, A. \tag{10}$$

**Discussion.** Although the retrieval of top-k images and the set union operation on their indices are not differentiable, the encoder transformer weights can still be updated by the gradient backpropagation from this loss function. In other words, we use the nearest neighbor indices to obtain the associated embeddings from the database, compute the similarities between these selected images, and pass a gradient back to the encoder.

In a sense, we treat the embeddings of top-k database images as constants. However, since they change with each query, the gradient is informed by the dot product computed between the query views and their top-k database images. Since the encoder weights are updated after each minibatch, all query views in the next minibatch, which go through a forward pass, have new embeddings. In contrast, the database images get new embeddings every few epochs (15 epochs in our implementation). See supplementary materials for more details.

The key benefit of approximate retrieval is the ability to scale to much larger database sizes as is common in industrial settings. As our experiments in Section 4.3 demonstrate, the performance in comparison to full retrieval is competetive.

## 3.5 Loss over Query Views and Database

Here we describe our loss formulation for a given input embedding and its views. There are two main ideas behind our loss design: divergence minimization and cross-attention based classification. Both of these losses compute a representation for the query conditioned on the database. In Section 4.3 we present a thorough ablation on the relative importance of each term in our loss function and also provide a discussion detailing the behavior that each term induces in the training process.

**Frobenius Loss for Divergence Minimization** Instead of directly minimizing the distance between $A$ views of the query, we propose to minimize the divergence between the distributions over all pairwise views conditioned over the database. Our intuition is that different views of the same query should retrieve similar database images. As described in Section 3.4, we interpret a softmax normalized similarities $P'$ as distributions, where each row gives a similarity distribution for each view over the retrieved image embeddings from the database. We compute a symmetric pairwise divergence matrix $L \in \mathbb{R}^{A \times A}$ using the Jensen-Shannon (JS) Divergence:

$$L_{ij} = \frac{KL\left(P'_i || \frac{(P'_i + P'_j)}{2}\right) + KL\left(P'_j || \frac{(P'_i + P'_j)}{2}\right)}{2}, \tag{11}$$

$$KL(P||R) = \sum_x P(x) \log\left(\frac{P(x)}{R(x)}\right), \tag{12}$$

where $P$ and $R$ are two distributions. Since different views of the same query should retrieve similar database images, each $L_{ij} \geq 0$ should be as small as possible. Hence, to compute an overall loss, we use the Frobenius Matrix Norm [Shen *et al.*, 2013] of matrix $L$ (using only the lower-triangular portion as the JS Divergence is symmetric):

$$\mathcal{L}_{frob} = \sqrt{\sum_{i=2}^{A} \sum_{j=1}^{i-1} L_{ij}^2}. \tag{13}$$

Intuitively each entry, $L_{ij}$, gives the distance between view $i$ and view $j$ conditioned on the distances to the retrieved database embeddings. So by minimizing $\mathcal{L}_{frob}$ we want new embeddings of views $i$ and $j$ to retrieve the same set of $k$-nearest neighbors with corresponding similarity values being similar. We reiterate that this loss is computed in a self-supervised way.

**Classification Loss** We use two modes of classification loss. The first computes a simple cross-entropy for each view embedding, $\mathbf{z}$. Note that the original query is included as one of the views. Each embedding is passed through a linear layer and softmax layer to produce an output distribution, $q$ over $c$ classes. This is then passed through the cross-entropy loss with labels $\mathbf{y}$.

$$q = \sigma(W_q \cdot \mathbf{z}) \in \mathbb{R}^{1 \times c} \tag{14}$$

$$\mathcal{L}_{ce} = -\sum_{m=1}^{c} y_c \log(q_m) \tag{15}$$

**CAC Loss** The second approach is to classify a new representation of the input query embedding, $\mathbf{z}$, which is obtained via cross-attention over the retrieved database embeddings, $Z_S^{union} \in \mathbb{R}^{T \times F}$, where $T$ is the total number of retrieved embeddings. We call this the *cross-attention classification* (CAC) loss. Intuitively the goal is to determine the class of $\mathbf{z}$ by attending on similar examples from the database and this setting closely resembles the category-level retrieval task where an input image of class $c_i$ returns other images of the same class from the database. We design our loss as follows

$$Q, K, V = W_q\mathbf{z}, W_k Z_S^{union}, W_v Z_S^{union} \tag{16}$$

$$\mathbf{z}' = \sigma(\frac{Q \cdot K^T}{\sqrt{F}})V \tag{17}$$

$$q = \sigma(W_c \cdot \mathbf{z}') \in \mathbb{R}^{1 \times c} \tag{18}$$

$$\mathcal{L}_{cac} = -\sum_{m=1}^{c} y_c \log(q_m), \tag{19}$$

where $W_q, W_k, W_v \in \mathbb{R}^{F \times F}$ and $W_c \in \mathbb{R}^{F \times c}$ for c classes. Note that both $\mathcal{L}_{ce}$ and $\mathcal{L}_{cac}$ are computed and summed over all query views $\mathbf{z}$.

**Overall Loss** Our combined loss is a simple weighted sum over each of the individual losses.

$$\mathcal{L}_{total} = \beta_{frob}\mathcal{L}_{frob} + \beta_{ce}\mathcal{L}_{ce} + \beta_{cac}\mathcal{L}_{cac} \tag{20}$$

We note that this loss performs well without any careful weighting and by simply allowing equal contribution for each term. In Section 4.3, we show experiments with different configurations for $\beta$ including $\beta_i = 0$, which removes the corresponding term.

## 4 Experiments

We follow a commonly used protocol for training and evaluation [Ermolov *et al.*, 2022; El-Nouby *et al.*, 2021a] and compare our method to state-of-the-art approaches across three datasets. We also share our implementation details and a thorough set of ablations on our architecture, hyperparameters, and loss design.

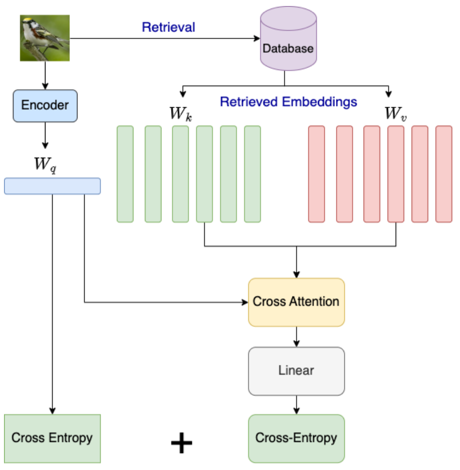

Figure 2: We show our Classification loss and Cross-Attention Classification (CAC) loss. The CAC loss computes cross-attention between a query embedding, $\mathbf{z}$ and the $k$-nearest neighbor embeddings from the database. This produces a new representation $\mathbf{z}'$ for $\mathbf{z}$ that is conditioned on the database. This can loosely be thought of as projecting the query to the basis produced by the database embeddings via a simple linear combination.

### 4.1 Datasets

In training, we use 80-20 stratified split over classes to create the Training Query and Training Database Sets, respectively. Our splits follow the common approaches used in [El-Nouby *et al.*, 2021a; Ermolov *et al.*, 2022].

**CUB-200** [Wah *et al.*, 2011] is a fine-grained dataset containing 11,788 images covering 200 sub-category classes of birds. We use the first 100 classes for training and the remaining for testing with no class overlap between train and test settings.

**In-Shop Clothes Retrieval** [Liu *et al.*, 2016] is a clothing dataset containing 52,712 total images and 7,896 classes. We use the first 3997 classes for training and the remaining classes for testing.

**Cars-196** [Krause *et al.*, 2013] is a dataset of car models containing 16,185 total images and 196 classes. We split the dataset into 8,054 images for training (98 classes) and the remaining 98 classes for testing.

**Stanford Online Products (SOP)** [Song *et al.*, 2016] contains 120,053 images of 22,634 products downloaded from eBay.com. We use the standard split of 59,551 images (11,318 classes) for training and 60,502 images (11,316 classes) for testing.

### 4.2 Implementation and Evaluation Details

We use two ViT variants as encoders across experiments, ViT-S [Dosovitskiy *et al.*, 2021] and DeiT-S [Touvron *et al.*, 2021] following other transformer-based retrieval approaches [Ermolov *et al.*, 2022; El-Nouby *et al.*, 2021a]. Both are based on the ViT-S architectures and are readily comparable across other convolutional approaches on parameter counts (ViT-S 22M, ResNet-50 23M [He *et al.*, 2015]). ViT-S is pretrained on ImageNet-21k [Deng *et al.*, 2009] which contains 14M images and 21K classes while DeiT-S is

| Method | Dim | Architecture | CUB-200 | | | | In-Shop | | | | Cars-196 | | | | Stanford Online Products | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 4 | 8 | 1 | 10 | 20 | 30 | 1 | 2 | 4 | 8 | 1 | 2 | 4 | 8 |
| NSoftmax [Zhai and Wu, 2019b] | 512 | R50 | 61.3 | 73.9 | 83.5 | 90 | 86.6 | 97.5 | 98.4 | 98.8 | 84.2 | 90.4 | 94.4 | 96.9 | 78.2 | 90.6 | 96.2 | - |
| ProxyNCA++ [Teh *et al.*, 2020b] | 512 | R50 | 69.0 | 79.8 | 87.3 | 92.7 | 90.4 | 98.1 | 98.8 | 99.0 | 86.5 | 92.5 | 95.7 | 97.7 | 80.7 | 92.0 | 96.7 | 98.9 |
| A-BIER [Opitz *et al.*, 2020] | 512 | GoogleNet | 57.5 | 68.7 | 78.3 | 86.2 | 93.1 | 95.1 | 96.9 | 97.5 | 82.0 | 89.0 | 93.2 | 96.1 | 74.2 | 86.9 | 94.0 | 97.8 |
| ABE [Kim *et al.*, 2018] | 512 | GoogleNet | 60.6 | 71.5 | 79.8 | 87.4 | 87.3 | 96.7 | 97.9 | 98.2 | 85.2 | 90.5 | 94.0 | 96.1 | 76.3 | 88.4 | 94.8 | 98.2 |
| SM [Suh *et al.*, 2019] | 512 | GoogleNet | 56.0 | 68.3 | 78.2 | 86.3 | 90.7 | 97.8 | 98.5 | 98.8 | 83.4 | 89.9 | 93.9 | 96.5 | 75.3 | 87.5 | 93.7 | 97.4 |
| Proxy-Anchor [Kim *et al.*, 2020b] | 512 | Inception-BN | 68.4 | 79.2 | 86.8 | 91.6 | 91.5 | 98.1 | 98.8 | 99.1 | 86.1 | 91.7 | 95.0 | 97.3 | 79.1 | 90.8 | 96.2 | 98.7 |
| SoftTriple [Qian *et al.*, 2019] | 512 | Inception-BN | 65.4 | 76.4 | 84.5 | 90.4 | - | - | - | - | 84.5 | 90.7 | 94.5 | 96.9 | 78.6 | 86.6 | 91.8 | 95.4 |
| HORDE [Jacob *et al.*, 2019] | 512 | Inception-BN | 66.8 | 77.4 | 85.1 | 91.0 | 90.4 | 97.8 | 98.4 | 98.7 | 86.2 | 91.9 | 95.1 | 97.2 | 80.1 | 91.3 | 96.2 | 98.7 |
| XBM [Wang *et al.*, 2020] | 512 | Inception-BN | 65.8 | 75.9 | 84.0 | 89.9 | 89.9 | 97.6 | 98.4 | 98.6 | 82.0 | 88.7 | 93.1 | 96.1 | 79.5 | 90.8 | 96.1 | 98.7 |
| MS [Wang *et al.*, 2019] | 512 | Inception-BN | 65.7 | 77.0 | 86.3 | 91.2 | 89.7 | 97.9 | 98.5 | 98.8 | 84.1 | 90.4 | 94.0 | 96.5 | 78.2 | 90.5 | 96.0 | 98.7 |
| HTL [Ge *et al.*, 2018] | 512 | Inception-BN | 57.1 | 68.8 | 78.7 | 86.5 | 80.9 | 94.3 | 95.8 | 97.2 | 81.4 | 88.0 | 92.7 | 95.7 | 74.8 | 88.3 | 94.8 | 98.4 |
| IRT_R [El-Nouby *et al.*, 2021b] | 384 | DeiT-S | 76.6 | 85.0 | 91.1 | 94.3 | 91.9 | 98.1 | 98.7 | 98.9 | - | - | - | - | 84.2 | 93.7 | 97.3 | 99.1 |
| Sph-DeiT [Ermolov *et al.*, 2022] | 384 | DeiT-S | 76.2 | 84.5 | 90.2 | 94.3 | 89.6 | 97.2 | 98.0 | 98.4 | 81.7 | 88.6 | 93.4 | 96.2 | 82.5 | 92.9 | 97.1 | 99.1 |
| Sph-DINO [Ermolov *et al.*, 2022] | 384 | ViT | 78.7 | 86.7 | 91.4 | 94.9 | 90.1 | 97.1 | 98.0 | 98.4 | 86.6 | 91.8 | 95.2 | 97.4 | 82.2 | 92.1 | 96.8 | 98.9 |
| Sph-ViT [Ermolov *et al.*, 2022] | 384 | ViT(IN21k) | 85.1 | 90.7 | 94.3 | 96.4 | 90.4 | 97.4 | 98.2 | 98.6 | 81.7 | 89.0 | 93.0 | 95.8 | 82.1 | 92.5 | 97.1 | 99.1 |
| Hyp-DeiT [Ermolov *et al.*, 2022] | 384 | DeiT-S | 77.8 | 86.6 | 91.9 | 95.1 | 90.5 | 97.8 | 98.5 | 98.9 | 86.4 | 92.2 | 95.5 | 97.5 | 83.3 | 93.5 | 97.4 | 99.1 |
| Hyp-DINO [Ermolov *et al.*, 2022] | 384 | ViT | 80.9 | 87.6 | 92.4 | 95.6 | 92.4 | 98.4 | **98.9** | 99.1 | **89.2** | **94.1** | **96.7** | **98.1** | 85.1 | 94.4 | 97.8 | 99.3 |
| Hyp-ViT [Ermolov *et al.*, 2022] | 384 | ViT(IN21k) | 85.6 | 91.4 | **94.8** | **96.7** | 92.5 | 98.3 | 98.8 | 99.1 | 86.5 | 92.1 | 95.3 | 97.3 | 85.9 | 94.9 | **98.1** | **99.5** |
| DMCAC-DeIT | 384 | DeiT-S | 78.4 | 87.0 | 92.3 | 95.0 | 91.1 | **98.5** | 98.8 | 99.1 | 84.4 | 89.2 | 94.9 | 97.5 | 84.2 | 93.6 | 97.4 | 99.1 |
| DMCAC-ViT | 384 | ViT (IN21k) | **86.2** | **92.0** | 94.7 | **96.7** | **92.7** | 98.2 | **98.9** | **99.3** | 88.5 | 93.9 | **96.7** | 98.0 | **86.3** | **95.2** | 97.5 | **99.5** |

Table 1: Recall@k metrics comparing across state-of-the-art methods on the CUB-200, In-Shop, Cars-196, and Stanford Online Products datasets. DMCAC (ours) performs competitively across architectures and outperforms all previous methods in several settings.

pretrained using the ImageNet-1k subset containing 1.3M images and 1K classes. DeiT is trained via distillation using a ResNet teacher model, more details can be found in [Touvron *et al.*, 2021]. Both encoders output embeddings of size 384. We use $224 \times 224$ input resolution across datasets. During training, we resize to 256 on the smaller size and take a random crop of size $224 \times 224$ while during testing we take a center crop of $224 \times 224$. We compute Recall@K as our evaluation metric during testing and use the embeddings $\mathbf{z}$ as described in Equation 5 projected to the unit ball. We choose RandAugment for query augmentation [Cubuk *et al.*, 2019b] across all experiments. This is chosen over learnable augmentation methods [Cubuk *et al.*, 2019a] for computational efficiency. Unless otherwise stated, we use $A = 6$ to generate $A$ views of a query (which includes the query as one of the views) during training and use the approximate retrieval approach with $k = 12$ rather than the full retrieval approach. There is no augmentation applied during testing. Unless otherwise stated, we use $\beta_{frob} = \beta_{ce} = \beta_{cac} = 1$ as described in Equation 20. We use AdamW [Loshchilov and Hutter, 2019] with learning rate $3 \times 10^{-5}$ as the optimizer with weight decay 0.01. We use batch size of 256 for all datasets except SOP where we use 128 and number of steps as 250, 600, 2500, and 25000 for CUB, Cars, In-Shop, and SOP respectively.

| Method | Betas | Recall@k | | | |
|---|---|---|---|---|---|
| | | 1 | 10 | 20 | 30 |
| Proxy-Anchor | - | 91.5 | 98.1 | 98.8 | 99.1 |
| Hyp-DINO | - | 92.4 | 98.4 | 98.9 | 99.1 |
| DMCAC-DeIT | [1,1,1] | 91.1 | 98.5 | 98.8 | 99.1 |
| DMCAC-ViT | | 92.7 | 98.2 | 98.9 | 99.3 |
| DMCAC-DeIT | [1,0,1] | 91.2 | 98.6 | 98.6 | 99.0 |
| DMCAC-ViT | | 92.7 | 98.3 | 98.8 | 99.4 |
| DMCAC-DeIT | [1,1,0] | 91.0 | 98.3 | 98.5 | 98.9 |
| DMCAC-ViT | | 92.4 | 98.3 | 98.7 | 99.3 |
| DMCAC-DeIT | [0.5,0.5,1] | 91.4 | 98.5 | 98.9 | 99.3 |
| DMCAC-ViT | | 93.0 | 98.4 | 98.9 | 99.5 |
| DMCAC-DeIT | [0.5,0,1] | 91.4 | 98.6 | 98.8 | 99.2 |
| DMCAC-ViT | | 93.1 | 98.5 | 98.9 | 99.4 |
| DMCAC-DeIT | [1,0.5,0.5] | 91.3 | 98.2 | 98.9 | 99.3 |
| DMCAC-ViT | | 92.9 | 98.5 | 98.8 | 99.4 |
| DMCAC-DeIT | [1,0.5,0] | 90.2 | 97.9 | 98.2 | 98.4 |
| DMCAC-ViT | | 91.9 | 96.9 | 97.2 | 97.8 |
| DMCAC-DeIT | [0,0.5,1] | 90.3 | 97.5 | 98.3 | 98.6 |
| DMCAC-ViT | | 91.7 | 96.9 | 97.9 | 98.1 |

Table 2: We compare the effect of tuning the loss weights Betas= $[\beta_{frob}, \beta_{ce}, \beta_{cac}]$ against the existing SOTA methods using both convolutional and ViT backbones on the In-Shop dataset. Overall we find that our base configuration of equally weighting terms already outperformed other methods making our method a good choice to use out-of-the-box, but there were minor gains made by careful tuning of these terms.

## 4.3 Results

We present results across a variety of settings and compare to other state-of-the-art methods across each of our datasets. In Table 1 we compare our method, DMCAC, against other state-of-the-art approaches that use both convolutional and ViT based backbones. Across the four datasets, the ViT based methods outperform their convolutional counterparts despite using a lower embedding dimension. For our DMCAC-DeiT method, we consistently outperform the other DeiT methods across all four datasets and are pushing the state-of-the-art results even against the ViT (ImageNet-21K) models which are pre-trained on a much larger dataset. Using our DMCAC-ViT method, we achieve several state-of-the-art results across the datasets and are comparable to existing ViT methods [Ermolov *et al.*, 2022]. We perform especially well on CUB-200, In-Shop, and Stanford Online Products. We hypothesize that this is because our approach can better distinguish between birds, clothing, and eBay products, which have more varied geometries and color patterns than the examples in Cars-196, though we are competitive there as well. We also make a similar observation to the one in [Ermolov *et al.*, 2022] that there is a large gap in performance between DMCAC-DeiT and DMCAC-ViT on CUB-200, which is likely because ImageNet-21k contains bird classes, which yields better separability for DMCAC-ViT.

| Views | DMCAC-DeIT | DMCAC-ViT |
|---|---|---|
| 3 | 78.6 | 86.0 |
| 6 | 78.4 | 86.2 |
| 9 | 78.3 | 86.3 |
| 12 | 78.2 | 86.1 |
| 15 | 77.9 | 86.1 |
| 20 | 77.7 | 86.0 |

Table 3: Ablation showing Recall@1 on CUB-200 when varying the number of views generated per query. The default choice is 6 views.

| | | Recall @ k | | | |
|---|---|---|---|---|---|
| Dataset | Method | 1 | 2 | 4 | 8 |
| CUB-200 | DMCAC-DeIT | 78.4 | 87.0 | 92.3 | 95.0 |
| | DMCAC-ViT | 86.2 | 92.0 | 94.7 | **96.7** |
| | DMCAC-DeIT - FR | 78.6 | 87.2 | 93.0 | 95.5 |
| | DMCAC-ViT - FR | **86.8** | **92.3** | **94.9** | **96.7** |
| Cars-196 | DMCAC-DeIT | 84.4 | 89.2 | 94.9 | 97.5 |
| | DMCAC-ViT | 88.5 | 93.9 | 96.7 | **98.1** |
| | DMCAC-DeIT - FR | 84.8 | 89.2 | 94.9 | 97.5 |
| | DMCAC-ViT - FR | **89.2** | **94.0** | **97.0** | 97.9 |

Table 4: We present an ablation of our approximate retrieval setting compared to the full retrieval setting (FR). We mostly see improvement in the FR setting compared to approximate retrieval, but the difference is relatively minor.

In Table 2 we present the effect of tuning our loss weights for each term in $\mathcal{L}_{total}$, namely $[\beta_{frob}, \beta_{ce}, \beta_{cac}]$. We highlight that our method works competitively out-of-the-box with the default configuration of equally weighting each loss term. This is clearly shown in Figure 3, where as training progresses, query examples learn to attend to the database examples of the same class. We find that with careful tuning of loss weights on the In-Shop dataset, we incrementally increase our already state-of-the-art retrieval performance.

We find that when we completely remove or down-weight $\mathcal{L}_{ce}$ relative to $\beta_{frob}$ and $\beta_{cac}$, retrieval performance stays stable compared to the baseline of equal weighting. However, if we remove either $\beta_{frob}$ or $\beta_{cac}$ as shown in the last two runs, performance drops drastically. This shows that both divergence minimization and cross-attention classification are necessary design choices in order to make our method work, and removing either significantly decrease retrieval performance. We hypothesize that this is because both terms have a synergistic relationship in the following way. As $\beta_{frob}$ decreases, the encoder learns both robustness to augmentations and to retrieve similar embeddings from the database across each augmented view. Simultaneously, the cross-attention operation used in $\beta_{cac}$ re-encodes each input view as a linear combination of the retrieved embeddings which are treated as *basis vectors*. A correct classification can then reasonably be achieved via a simple linear combination only if the retrieved embeddings capture the semantic meaning of the query. This is the very goal of category level retrieval.

In Table 3 we perform an ablation on the number of views generated per image and how this affects Recall@1 for CUB-200. There is a tradeoff between the computational complexity of generating more views and the benefits of learning invariances over a greater number of augmentations. The effective batch size increases proportionally to the number of generated views which can be limiting depending on available GPU memory. However, we find that retrieval results stayed stable across the DMCAC-ViT settings and degraded for more views when using DMCAC-DeiT. Thus we find that our method is relatively stable to the number of augmentations used and thus is accessible and performs well even in low-memory environments.

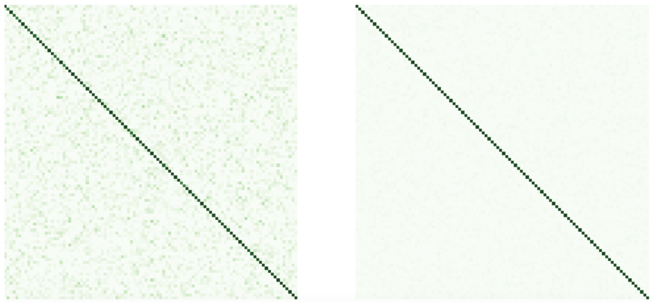
Figure 3: Cross-Attention maps between the validation query and database sets at iteration 5 (left) and 100 (right) for CUB-200 with DMCAC-ViT. Entry (i,j) shows the similarity of query of class $i$ to database of class $j$. As training progresses, queries attend to database examples of the same class shown by the diagonal line.

In Table 4 we compare our approximate retrieval setting against full retrieval (FR) on CUB-200 and Cars-196. Due to the increase in memory and computational complexity of FR we reduce the relative size of the database compared to the query set during training. The testing setting remains unchanged. For CUB-200 we use a stratified 90-10 query-database split during training and a stratified 95-5 split for Cars-196 during training. We see that the FR outperforms approximate retrieval across both datasets and models in most cases, but the difference is relatively minor. We hypothesize that this is because FR is end-to-end differentiable which means a gradient signal gets passed back to the encoder from all of the database embeddings rather than just from the subset of top-k examples. This acts as a sort of negative sampling where the encoder sees the full database distribution, including the long tail of low-similarity examples for a given query.

## 5 Conclusions

We offer a new perspective on image retrieval using self-supervision with a novel loss formulation. We hypothesize that our distribution matching objective learns better query representations than point-wise objectives which can suffer from sampling noise. We instead learn a full mapping from the queries to the database via $\mathcal{L}_{frob}$. We also hypothesize that cross-attention classification (CAC) is better aligned to retrieval than cross-entropy classification (CEC). In CAC, each query is projected to a "database space" as an attention-weighted linear combination of the database images and is correctly classified if its "database space" representation accurately captures its class.

## Acknowledgements

This work was supported in part by the NSF Grants III-2107213 and CNS-2333611.

## References

[Ba *et al.*, 2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[Baevski *et al.*, 2022] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language, 2022.

[Boudiaf *et al.*, 2021] Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses, 2021.

[Caron *et al.*, 2019] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features, 2019.

[Caron *et al.*, 2021a] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments, 2021.

[Caron *et al.*, 2021b] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021.

[Carreira-Perpiñán and Hinton, 2005] Miguel Á. Carreira-Perpiñán and Geoffrey Hinton. On contrastive divergence learning. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pages 33–40. PMLR, 06–08 Jan 2005. Reissued by PMLR on 30 March 2021.

[Chen *et al.*, 2020] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[Chen *et al.*, 2021] Chun-Fu (Richard) Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 357–366, October 2021.

[Cubuk *et al.*, 2019a] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data, 2019.

[Cubuk *et al.*, 2019b] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[Doersch *et al.*, 2016] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction, 2016.

[Dosovitskiy *et al.*, 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[El-Nouby *et al.*, 2021a] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval, 2021.

[El-Nouby *et al.*, 2021b] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval, 2021.

[Ermolov *et al.*, 2022] Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe, and Ivan Oseledets. Hyperbolic vision transformers: Combining improvements in metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7409–7419, June 2022.

[Ge *et al.*, 2018] Weifeng Ge, Weilin Huang, Dengke Dong, and Matthew R. Scott. Deep metric learning with hierarchical triplet loss, 2018.

[Hadsell *et al.*, 2006] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[He *et al.*, 2021] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021.

[Hoffer and Ailon, 2018] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network, 2018.

[Jacob *et al.*, 2019] Pierre Jacob, David Picard, Aymeric Histace, and Edouard Klein. Metric learning with horde: High-order regularizer for deep embeddings, 2019.

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

[Khosla *et al.*, 2021] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2021.

[Kim *et al.*, 2018] Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon. Attention-based ensemble for deep metric learning, 2018.

[Kim *et al.*, 2020a] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning, 2020.

[Kim *et al.*, 2020b] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[Koppula *et al.*, 2022] Skanda Koppula, Yazhe Li, Evan Shelhamer, Andrew Jaegle, Nikhil Parthasarathy, Relja Arandjelovic, João Carreira, and Olivier Hénaff. Where should i spend my flops? efficiency evaluations of visual pre-training methods, 2022.

[Krause *et al.*, 2013] 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.

[Liu *et al.*, 2016] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1096–1104, 2016.

[Liu *et al.*, 2021] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[Loshchilov and Hutter, 2019] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[Movshovitz-Attias *et al.*, 2017a] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 360–368, Los Alamitos, CA, USA, oct 2017. IEEE Computer Society.

[Movshovitz-Attias *et al.*, 2017b] Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies, 2017.

[Noroozi and Favaro, 2017a] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles, 2017.

[Noroozi and Favaro, 2017b] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles, 2017.

[Opitz *et al.*, 2020] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):276–290, 2020.

[Park *et al.*, 2023] Namuk Park, Wonjae Kim, Byeongho Heo, Taekyung Kim, and Sangdoo Yun. What do self-supervised vision transformers learn?, 2023.

[Qian *et al.*, 2019] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[Robinson *et al.*, 2021] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples, 2021.

[Shen *et al.*, 2013] Chunhua Shen, Junae Kim, Fayao Liu, Lei Wang, and Anton van den Hengel. An efficient dual approach to distance metric learning, 2013.

[Song *et al.*, 2016] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[Song *et al.*, 2023] Chull Hwan Song, Jooyoung Yoon, Shunghyun Choi, and Yannis Avrithis. Boosting vision transformers for image retrieval. In *WACV*, 2023.

[Steiner *et al.*, 2022] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers, 2022.

[Strudel *et al.*, 2021] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation, 2021.

[Suh *et al.*, 2019] Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. Stochastic class-based hard example mining for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[Szegedy *et al.*, 2014] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[Teh *et al.*, 2020a] Eu Wern Teh, Terrance DeVries, and Graham W. Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis, 2020.

[Teh *et al.*, 2020b] Eu Wern Teh, Terrance DeVries, and Graham W. Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis, 2020.

[Touvron *et al.*, 2021] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers distillation through attention, 2021.

[Trivedy and Latecki, 2023] Vivek Trivedy and Longin Jan Latecki. Cnn2graph: Building graphs for image classification. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1–11, 2023.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[Wah *et al.*, 2011] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Caltech-ucsd birds-200-2011. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[Wang *et al.*, 2019] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[Wang *et al.*, 2020] Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R. Scott. Cross-batch memory for embedding learning, 2020.

[Yu *et al.*, 2019] Jian Yu, Changhui Hu, Xiaoyuan Jing, Guangliang Zhou, and Shen Jing. Deep metric learning with dynamic margin hard sampling loss. In *2019 Chinese Control Conference (CCC)*, pages 7901–7905, 2019.

[Zhai and Wu, 2019a] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning, 2019.

[Zhai and Wu, 2019b] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning, 2019.