Asymmetric metric learning for knowledge transfer

Mateusz Budnik  Yannis Avrithis
Inria, Univ Rennes, CNRS, IRISA

Abstract

Knowledge transfer from large teacher models to smaller student models has recently been studied for metric learning, focusing on fine-grained classification. In this work, focusing on instance-level image retrieval, we study an asymmetric testing task, where the database is represented by the teacher and queries by the student. Inspired by this task, we introduce asymmetric metric learning, a novel paradigm of using asymmetric representations at training. This acts as a simple combination of knowledge transfer with the original metric learning task.

We systematically evaluate different teacher and student models, metric learning and knowledge transfer loss functions on the new asymmetric testing as well as the standard symmetric testing task, where database and queries are represented by the same model. We find that plain regression is surprisingly effective compared to more complex knowledge transfer mechanisms, working best in asymmetric testing. Interestingly, our asymmetric metric learning approach works best in symmetric testing, allowing the student to even outperform the teacher.

Our implementation is publicly available, including trained student models for all loss functions and all pairs of teacher/student models.

1. Introduction

Originating in metric learning, loss functions based on pairwise distances or similarities [18, 71, 46, 72, 5] are paramount in representation learning. Their power is most notable in category-level tasks where classes at inference are different than classes at learning, for instance fine-grained classification [46, 72], few-shot learning [69, 62] local descriptor learning [19] and instance-level retrieval [16, 53]. There are different ways to use them without supervision [31, 80, 6] and indeed, they form the basis for modern unsupervised representation learning [44, 21, 8].

Powerful representations come traditionally with powerful network models [22, 28], which are expensive. The search for resource-efficient architectures has lead to the design of lightweight networks for mobile devices [26, 58, 84], neural architecture search [48, 41] and model scaling [64]. Training of small networks may be facilitated by knowledge transfer from larger networks [23]. However, both network design and knowledge transfer are commonly performed on classification tasks, using standard cross-entropy.

Focusing on fine-grained classification and retrieval, several recent methods have extended metric learning loss functions to allow for knowledge transfer from teacher to student models [9, 38, 82, 47]. However, two questions are in order: (a) since transferring a representation from one model to another is inherently a continuous task, can’t we just use regression? (b) apart from knowledge transfer, is the original metric learning task still relevant and what is a simple way to combine the two?

In this work, we focus on the task of instance-level image retrieval [49, 52], which is at the core of metric learning in the sense of using pairwise distances or similarities. In its most well-known form [16, 53], the task is supervised, but the supervision is originating from automated data analysis rather than humans. As such, apart from noisy, supervision is often incomplete, in the sense that although class labels per example may exist, not all pairs of examples of the same class are labeled. Hence, one has to work with pairs rather than examples, unlike e.g. face recognition [11].

Our work is motivated by the scenario where a database (gallery) of images is represented and indexed according to a large model, while queries are captured from mobile devices, where a smaller model is the only option. In such scenario, rather than re-indexing the entire database, it is preferable to adapt different smaller models for different end-user devices. In this case, knowledge transfer from the large (teacher) to the small (student) model is not just helping, but the student should really learn to map inputs to the same representation space. We call this task asymmetric testing.

More importantly, even if we consider the standard symmetric testing task, where both queries and database examples are represented by the same model at inference, we

---

1 https://github.com/budnikm/aml
2 This work was supported by Conseil Régional de Bretagne and Rennes Métropole (Images & Réseaux AAP-PME-2017 grant MobiliAI). It was performed using HPC resources of GENCI-IDRIS (Grant 2019-AD011011245).
introduce a novel paradigm of using asymmetric representations at training, as a knowledge transfer mechanism. We call this paradigm asymmetric metric learning. By representing anchors by the student and positives/negatives by the teacher, one can apply any metric learning loss function. This achieves both metric learning and knowledge transfer, without resorting to a linear combination of two loss functions. In summary, we make the following contributions:

• We study the problem of knowledge transfer from a teacher to a student model for the first time in pair-based metric learning for instance-level image retrieval.

• In this context, we study the asymmetric testing task, where the database is represented by the teacher and queries by the student.

• In both symmetric and asymmetric testing, we systematically evaluate different teacher and student models, metric learning loss functions (subsection 3.3) and knowledge transfer loss functions (subsection 3.4), serving as a benchmark for future work.

• We introduce the asymmetric metric learning paradigm, an extremely simple mechanism to combine metric learning with knowledge transfer (subsection 3.2).

2. Related work

Metric learning Historically, metric learning is about unsupervised learning of embeddings according to a pairwise distances [65] or similarities [59, 3]. Modern deep metric learning is mostly supervised, with pair labels specifying a set of positive and negative examples per anchor example [78]. Standard loss functions are contrastive [18] and triplet [78, 71], operating on one or two pairs, respectively. Global loss functions rather operate on an arbitrary number of pairs [46, 72, 5], similarly to learning to rank [7, 77]. The large number of potential tuples gives rise to mining [20, 75] and memory [73, 76] mechanisms. At the other extreme, extensions of cross-entropy operate on single examples [70, 11]. We focus on pair-based functions in this work, due to the nature of the ground truth [53, 54]. Unsupervised metric learning is gaining momentum [31, 80, 6], but we focus on the supervised case, given that it requires no human effort [53, 16].

Image retrieval Instance-level image retrieval, either using local features [66] or global pooling [36], has relied on SIFT descriptors [43] for more than a decade. Convolutional networks quickly outperformed shallow representations, using different pooling mechanisms [56, 68] and fine-tuning on relevant datasets, initially with cross-entropy on noisy labels from the web [2] and then with contrastive [53] and triplet [16] loss on labels generated from the visual data alone. While the best performance comes from large networks [22, 52], we focus on small networks [58, 64] for the first time. Our asymmetric test scenario is equivalent to that of prior studies [27, 60], but with different motivation and settings. Feature translation [27] is meant for retrieval system interoperability, so both networks may be large and none is adapted. The recent backward-compatible training (BCT) [60] is meant to avoid re-indexing of the database like here, but the new model used for queries is actually more powerful than the old one used for the database, or trained on more data.

Small networks While large networks [22, 28] excel in performance, they are expensive. One solution is to compress existing architectures, e.g. by quantization [15] or pruning [42]. Another is to manually design more efficient networks, e.g. by using bottlenecks [30], separable convolutions [26], inverted residuals [58] or point-wise group convolutions [84]. MobileNetV2 [58] is such a network that we use as a student in this work. More recently, neural architecture search [48, 41, 63, 25] is making this process automatic, although expensive. Alternatively, a small model can first be designed (or learned) and then its architecture scaled by adding depth [22], width [83], resolution [29] or a compound of the above [64]. We use the latter as another student in this work. We show that pruning [74] cannot compete designed or learned architectures.

Knowledge transfer Rather than training a small network directly, it is easier to optimize the same small network (student) to mimic a larger one (teacher), essentially transferring knowledge from the teacher to the student. In classification, this can be done e.g. by regression of the logits [1] or by cross-entropy on soft targets, known as knowledge distillation [23]. BCT [60] fixes the classifier (last layer) of the student to that of the teacher, similarly to [24]. Such ideas do not apply in this work, since there is no parametric classifier. Metric learning is mostly about pairs rather than individual examples, and indeed recent knowledge transfer methods are based on pairwise distances or similarities. This includes e.g. learning to rank [9] and regression on quantities involving one or more pairs like distances [82, 47], log-ratio of distances [38], or angles [47]. The most general form is relational knowledge distillation (RKD) [47]. Direct regression on features is either not considered or shown inferior [82], but we show it is much more effective than previously thought. We also show that the original metric learning task is still beneficial when training the student and we combine with knowledge transfer in a simple way.

Asymmetry Asymmetric distances or similarities are common in approximate nearest neighbor search, where queries may be quantized differently than the database, or not at
all [12, 17, 34, 32, 45, 10]. In image retrieval, there are efforts to reduce the asymmetry of k-nearest neighbor relations [35], or use asymmetry to mitigate the effect of quantization [33], or handle partial similarity [85] or alignment [67]. In classification, it is common to use asymmetric image-to-class distances [4] or region-to-image matching [37]. In metric learning, asymmetry has been used in sample weighting [40], different mappings per view [81], or hard example mining [79]. Asymmetric similarities are used between cross-modal embeddings [14, 39, 13], but not for knowledge transfer. They are also used over the same modality to adjust embeddings to a memory bank [21] or to treat a set of examples as a whole [69], but again not for knowledge transfer.

3. Asymmetric metric learning

3.1. Preliminaries

Let \( X \subset \mathcal{X} \) be a training set, where \( \mathcal{X} \) is an input space. Two sources of supervision are considered. The first is a set of labels: a subset of all pairs of examples in \( X \) is labeled as positive or negative and the remaining are unlabeled. Formally, for each anchor \( a \in X \), a set \( P(a) \subset X \) of positive and a set \( N(a) \subset X \) of negative examples are given. The second is a teacher model \( g : \mathcal{X} \to \mathbb{R}^d \), mapping input examples to a feature (embedding) space of dimensionality \( d \). The objective is to learn the parameters \( \theta \) of a student model \( f_\theta : \mathcal{X} \to \mathbb{R}^d \), such that anchors are closer to positives than negatives, the teacher and student agree in some sense, or both. When labels are not used, an additional set of examples \( \mathcal{U}(a) \) may be used for each anchor \( a \), e.g. a neighborhood of space or the entire set \( X \setminus \{a\} \). The teacher is assumed to have been trained on \( X \) using labels only.

Training amounts to minimizing the error function

\[
J(X; \theta) := \sum_{a \in X} \ell(a; \theta)
\]

with respect to parameters \( \theta \) over \( X \). There is one loss term per anchor \( a \in X \), which however may depend on any other example in \( X \); hence, \( J \) is not additive in \( X \). The loss function \( \ell \) may depend on the labels or the teacher only, discussed respectively in subsection 3.3 and subsection 3.4; it may depend on the teacher indirectly via a similarity function, as discussed in subsection 3.2.

At inference, a test set \( Z \subset \mathcal{X} \) and a set of queries \( Q \subset \mathcal{X} \) are given, both disjoint from \( X \). For each query \( q \in Q \), a set \( P(q) \subset Z \) of positive examples is given. Symmetric testing is the task of ranking positive examples \( P(q) \) before all others in \( Z \) by descending similarity to \( q \) in the student space, for each query \( q \in Q \). Asymmetric testing is the same, except that similarities are between queries in the student space and test examples in the teacher space.

3.2. Asymmetric similarity

We use cosine similarity in this work: \( \text{sim}(v, v') := \langle v, v' \rangle / (\|v\| \cdot \|v'\|) \) for \( v, v' \in \mathbb{R}^d \). The symmetric similarity \( s_\theta^{\text{sym}}(a, x) \) between an anchor \( a \in X \) and a positive or negative example \( x \in P(a) \cup N(a) \) is obtained by representing both in the feature space of the student:

\[
s_\theta^{\text{sym}}(a, x) := \text{sim}(f_\theta(a), f_\theta(x)).
\]

This is the standard setting in related work in metric learning.

By contrast, we introduce the asymmetric similarity \( s_\theta^{\text{sym}}(a, x) \), where the anchor \( a \) is represented by the stu-
called DarkRank [9] by \( \psi(a, x) := \|a - x\| \) and the direct match baseline of DarkRank [9] by \( \psi(a, x) := \|a - x\|^2 \).

**DarkRank** Let \( V(a, x) := \{ y \in U(a) : S(a, y) \leq S(a, x) \} \) be the set of examples in \( U(a) \) that are mapped farther away from anchor \( a \) than \( x \) in the teacher space. For each \( x \in U(a) \), DarkRank [9] encourages those examples to be farther away from \( a \) than \( x \) in the student space:

\[
\ell_{\text{DR}}(a; \theta) := -\sum_{x \in U(a)} \left( s_{\theta}^{\text{sym}}(a, x) - \log \sum_{y \in V(a, x)} e^{s_{\theta}^{\text{sym}}(a, y)} \right),
\]

**3.4. Loss functions using the teacher only**

When not using the labels, the only source of supervision is the teacher model \( g \). Symmetric similarity (2) is not an option here; we either use use (3) or other ways to compare the two models. Given anchor \( a \), the loss may depend on \( a \) alone, or also the additional examples \( U(a) \). We write \( S(a, x) := \sim(a, g(x)) \) for the similarity of \( a \) and some \( x \in U(a) \) in the teacher space.

**Regression** The simplest option is regression, encouraging the representations of the same input example \( a \) by the two models to be close by using asymmetric similarity (3):

\[
\ell_{\text{R}}(a; \theta) := -s_{\theta}^{\text{asym}}(a, a) = -\sim(a, g(a)).
\]

For each anchor, it does not depend on any other example. It is the same as the absolute version of metric learning knowledge distillation (ML+KD) [82] and as contrastive loss (4) on asymmetric similarity (3) (using only the anchor as a positive for itself).

**Relational distillation** Given an anchor \( a \) and one or more other vectors \( x, \ldots \in \mathbb{R}^d \), relational knowledge distillation (RKD) [47] is based on a number of relational measurements \( \psi(a, x, \ldots) \). One such \( \psi(a, x, \ldots) \) is the distance \( \|a - x\| \) for \( x \in \mathbb{R}^d \). Another is the angle \( \sim(a - x, a - y) \) formed by \( a, x, y \), for \( x, y \in \mathbb{R}^d \). The loss is called distance-wise and angle-wise, respectively. The RKD loss encourages the same measurements by both models,

\[
\ell_{\text{RKD}}(a; \theta) := \sum_{(x, \ldots) \in U(a)^n} r(\psi(f_\theta(a), f_\theta(x), \ldots), \psi(g(a), g(x), \ldots)),
\]

where \( n \) is e.g. 1 for distance and 2 for angle and \( r \) is a regression loss, taken as Huber [47]. RKD encompasses regression by \( \psi \) taken as the identity mapping on the anchor feature alone and \( r \) taken as \( -\sim \). It also encompasses the relative setting of ML+KD [82] by \( \psi(a, x) := \|a - x\| \) and the direct match baseline of DarkRank [9] by \( \psi(a, x) := \|a - x\|^2 \).
We use the revisited (GeM) on the last convolutional feature map. (mP@10) results are included in the supplementary material.

Table 1 gives the number of parameters and computational complexity (in FLOPS) for the networks used in this work. It is important to note that the versions of the teachers are already adapted for image retrieval by removing the fully connected layers, hence our version of VGG16 has significantly fewer parameters than the original (around 138M). Student networks are adapted in the same way, that is, fully connected layers removed. In addition, each student is shown with or without the added layer (or not) to match the output dimensionality of the teacher.

4. Experiments

4.1. Setup

Datasets We use the SfM dataset [54] for training, containing 133k images for training and 30k images for validation. We use the revisited R Oxford5k and R Paris6k datasets [52] for testing, each having 70 query images. All datasets depict particular architectural landmarks under very diverse viewing conditions. We follow the standard evaluation protocol, using the medium and hard settings [52]. We report mean average precision (mAP) here, while mean precision at 10 (mP@10) results are included in the supplementary material. Comparisons are based on mAP. Also included in the supplementary material are additional symmetric and asymmetric testing results for both datasets and both metrics in the presence of 1M distractors [52], denoted as R1M. To compare with pruning [74], we also use the original Oxford5k [49] and Paris6k [50] datasets, reporting mAP only.

Networks All models are pre-trained for classification on ImageNet [57] and then fine-tuned for image retrieval on SfM, following the setup of the same work. We use VGG-16 [61] and ResNet101 [22] as teacher models with the feature dimensionality d of 512 and 2048, respectively. We use MobileNetV2 [58] and EfficientNet-B3 [64] as student networks, removing any fully connected layers and stacking one 1 × 1 convolutional layer to match the dimensionality of the teacher. All networks use generalized mean-pooling (GeM) [54] on the last convolutional feature map.

Complexity and parameters Table 1 gives the number of parameters and computational complexity (in FLOPS) for the networks used in this work. It is important to note that the versions of the teachers are already adapted for image retrieval by removing the fully connected layers, hence our version of VGG16 has significantly fewer parameters than the original (around 138M). Student networks are adapted in the same way, that is, fully connected layers removed. In addition, each student is shown with or without the 1 × 1 convolutional layer per teacher.

Implementation details The image resolution is limited to 362 × 362 at training (fine-tuning). At testing, a multi-scale representation is used, with initial resolution of 1024 × 1024 and scale factors of 1, 1/√2 and 1/2. The representation is pooled by GeM over the features of the three scaled inputs. We use supervised whitening, trained on the same SfM dataset [54]. In asymmetric testing, whitening is learned in the teacher space. Our implementation is based on the official code of [54] in PyTorch [51], as well as [72, 47, 9]. Teacher models are taken from [54]. Our implementation is publicly available, including trained student models for all loss functions and all pairs of teacher/student models.

Training and hyper-parameters We follow the training setup of [54] for loss functions that use labels. We use the validation set to determine the hyperparameter values and the best model. We train all models using the SGD with learning rate decay of 0.99 per epoch. Symmetric training (2) takes place for 100 epochs or until convergence based on the validation set. For asymmetric training (3), this is extended to 300 epochs. Each epoch consists of 2000 tuples. A mini-batch has 10 tuples, each composed of 1 anchor, 1 corresponding positive and 5 negatives. For unsupervised losses we create tuples of the same overall size. We use weight decay of 10−6 in each experiment.

3https://github.com/budnikm/aml
We use the DA variant of RKD [47] (8), with the angle-wise and distance-wise loss weighted by a factor of 2 and 

\[ \eta = 10^{-6} \] for symmetric and asymmetric training, respectively. For triplet (5), we set \( m = 0.1 \) and \( \eta = 10^{-8} \). For multi-similarity (6) we set \( m = 0.6, \alpha = 1, \beta = 1 \) and \( \eta = 10^{-8} \) for all setups. For regression (7), we set \( \eta = 10^{-3} \). We use the DA variant or RKD [47] (8), with the angle-wise and distance-wise loss weighted by a factor of 2 and 1 respectively, and \( \eta = 10^{-2} \). For DarkRank (DR) (9), we set \( \eta = 10^{-6} \) for the VGG16 teacher; for ResNet101, \( \eta = 10^{-5} \) for MobileNetV2 and \( \eta = 10^{-7} \) for EfficientNetB3. We do not discriminate between the student training being supervised or not, since labels are already used for teacher training.

**Mining** When using labels, we use hard negative mining as a default, following [54]. Negatives are mined each epoch from a random subset of 22k images of the training set. The negatives closest to the anchor (according to (2) or (3), depending on the setting) are selected. There is no mining for positives, because there are only few (1-2) positives per anchor. When not using labels, we draw additional examples uniformly at random as a default. There is no mining for regression.

**Example of asymmetric hard example mining** Depending on the similarity we use in the loss function, i.e., symmetric (2) or asymmetric (3), we follow the same choice for hard negative mining. This means that, in mining based on asymmetric similarity, the features of anchor \( a \) come from the student model \( f_o(a) \), while the database is represented by the teacher \( g \). The database is fixed and does not need to be re-computed after each epoch. Only the anchors are updated, which makes training more efficient.

Figure 2 gives an example of the hard negatives mined for one anchor across different epochs. Before the training starts (epoch 0) the results are not very informative, which is not surprising giving that the feature spaces of the teacher and the student do not match. However, after just a few epochs we see harder negatives being selected. This example illustrates how asymmetric similarity acts as a knowledge transfer mechanism from the teacher to the student model.

**4.2. Results**

**Contrastive-regression ablation** As will be shown in the following results, contrastive loss and regression turn out to be most effective in general. Moreover, by comparing (4) with (7), contrastive with asymmetric similarity (3) encompasses regression by setting each anchor as a positive for itself, without any other positive or negatives. To better understand the relation between these two loss functions, we perform an ablation study where we investigate versions of contrastive on (3) having negatives, or not, and the anchor itself as positive, or not.

The results are shown in Table 2 for the case of VGG16→MobileNetV2. In symmetric testing, it turns out that the best combination is having both negatives and the anchor itself. The same happens in almost all cases for other teacher and student models, as shown in the supplementary material. We denote this combination as Contr\(^+\) and we include it in subsequent results:

\[
\ell_{\text{C}+} (a; \theta) := \sum_{n \in N(a)} [s_{\theta}(a, n) - m]_+ - \sum_{p \in P(a)} s_{\theta}(a, p) - s_{\theta}(a, a), \tag{10}
\]

where \( s_{\theta} \) is asymmetric (3). Asymmetric testing is much more challenging. The best is regression in this case, but Contr\(^+\) is still the second best.

**Symmetric testing** According to the left part of Table 3, Contr\(^+\) works best on MobileNetV2, while on EfficientNet, either contrastive and Contr\(^+\) works best, with the two options having little difference. The difference to other loss functions using labels is large, reaching 20% or even 30% on ROxford5k. Triplet is known to be inferior to contrastive [54], but the difference is more pronounced in our knowledge transfer setting. This result is particularly surprising for multi-similarity, which is state of the art in fine-grained classification [72]. Also surprisingly, regression works best among loss functions not using labels, including

<table>
<thead>
<tr>
<th>Student</th>
<th>Teacher</th>
<th>Lab</th>
<th>Loss</th>
<th>Self Pos Neg Mining</th>
<th>Asym</th>
<th>Symmetric Testing</th>
<th>Asymmetric Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R\text{Oxford}5k</td>
<td>R\text{Paris}6k</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ Contr (4) ✓ ✓ hard ✓</td>
<td>57.3</td>
<td>67.1</td>
<td>31.1</td>
<td>41.3</td>
<td>38.3</td>
<td>49.8</td>
<td>18.4</td>
</tr>
<tr>
<td>✓ Contr (4) ✓ ✓ hard ✓</td>
<td>57.3</td>
<td>68.4</td>
<td>31.5</td>
<td>42.2</td>
<td>42.9</td>
<td>55.9</td>
<td>22.6</td>
</tr>
<tr>
<td>MobileNetV2 VGG16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ Contr (4) ✓ ✓ hard ✓</td>
<td>55.9</td>
<td>66.7</td>
<td>31.1</td>
<td>40.6</td>
<td>34.1</td>
<td>47.3</td>
<td>17.0</td>
</tr>
<tr>
<td>✓ Contr (4) ✓ ✓ hard ✓</td>
<td>55.5</td>
<td>67.0</td>
<td>30.4</td>
<td>40.9</td>
<td>38.2</td>
<td>52.2</td>
<td>15.3</td>
</tr>
<tr>
<td>Reg (7) ✓ – ✓</td>
<td>53.3</td>
<td>67.5</td>
<td>28.9</td>
<td>40.9</td>
<td>48.0</td>
<td>57.9</td>
<td>26.5</td>
</tr>
</tbody>
</table>

Table 2. Contrastive-regression ablation. Symmetric and asymmetric testing mAP on ROxford5k and RParis6k [52]. Lab: using labels in student model training. Pos, Neg: Using positives, negatives. Self: Using anchor (by teacher) as positive for itself (by student). Asym: Using asymmetric similarity (3) at training. The second row is Contr\(^+\) (10). GeM pooling and learned whitening [54] used in all cases.

Loss functions For contrastive loss (4), we set the margin \( m = 0.7 \) and the initial learning rate \( \eta \) to \( 10^{-5} \) and \( 10^{-3} \) for symmetric and asymmetric training, respectively. For triplet (5), we set \( m = 0.1 \) and \( \eta = 10^{-8} \). For multi-similarity (6) we set \( m = 0.6, \alpha = 1, \beta = 1 \) and \( \eta = 10^{-8} \) for all setups. For regression (7), we set \( \eta = 10^{-3} \).
recent knowledge transfer methods RKD [47] and DarkRank [9]. It is second or third best in all cases. This finding is contrary to [82], where the regression baseline is found inferior. DarkRank is inferior to RKD, in agreement with [47].

The superiority of contrastive or Contr+ over regression confirms that, by using our asymmetric similarity, the original metric learning task is still beneficial. Unlike knowledge distillation on classification tasks, knowledge transfer alone is not the best option. Focusing on the best results (contrastive or Contr+), we confirm that, with just one exception (VGG16→EfficientNet on /Paris6k hard), knowledge transfer always helps compared to training without the teacher, using the same \( d \). The gain is more pronounced, reaching 7-10% on ResNet101→MobileNetV2, when the teacher is stronger and the student is weaker (the exception corresponds to the weakest teacher and strongest student). MobileNetV2 performs only 2-3% below its teacher. Remarkably, EfficientNet outperforms its teacher: this happens on /Paris6k for VGG16 and on all settings for ResNet101. This is also the case in the presence of /1M distractors for ResNet101 teacher. This result can be found in the supplement, where symmetric testing results for all losses and models are included on /Oxford5k +/Paris6k. Lab: using labels in student model training. Asym: Using asymmetric similarity (3) at training (our work). Best result highlighted per teacher-student pair. GeM pooling and learned whitening [54] used in all cases.

### Table 3. Symmetric and asymmetric testing mAP on /Oxford5k and /Paris6k [52]. Lab: using labels in student model training. Asym: Using asymmetric similarity (3) at training (our work). Best result highlighted per teacher-student pair. GeM pooling and learned whitening [54] used in all cases.

<table>
<thead>
<tr>
<th>Student</th>
<th>Teacher</th>
<th>Loss</th>
<th>Mining</th>
<th>Asym</th>
<th>Symmetric Testing</th>
<th>Asymmetric Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( d )</td>
<td></td>
<td></td>
<td>Medium</td>
<td>Hard</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ROxf</td>
<td>RPar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ROxf</td>
<td>RPar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ROxf</td>
<td>RPar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ROxf</td>
<td>RPar</td>
</tr>
</tbody>
</table>

**Asymmetric testing** Here, similarities are asymmetric at testing, with the database being represented by the teacher and queries by the student. According to the right part of Table 3, regression is the clear winner in this case. This is contrary to [60], where regression fails. In a sense, this can be expected since the student should learn to map images...
Table 4. Symmetric testing mAP on Paris6k and Oxford5k. FLOPS and parameters relative to VGG16. LAB: using labels in student model training. Using asymmetric similarity (3) in all teacher-student settings. GeM pooling [54] used in all cases but not learned whitening.

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>%FLOPS</th>
<th>%PARAM</th>
<th>TEACHER</th>
<th>LAB</th>
<th>LOSS</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>100</td>
<td>100</td>
<td>✓ Contr (4)</td>
<td>82.45 81.37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VGG16-PLFP [74]</td>
<td>57.37</td>
<td>61.05</td>
<td>✓ Contr (4)</td>
<td>76.20 73.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MobileNetV2</td>
<td>2.44</td>
<td>19.58</td>
<td>VGG16</td>
<td>✓ Contr (4) 74.14 78.26 Reg (7)</td>
<td>66.58 74.45</td>
<td></td>
</tr>
<tr>
<td>MobileNetV2</td>
<td>3.14</td>
<td>32.97</td>
<td>ResNet101</td>
<td>✓ Contr (4) 75.30 83.23 Reg (7)</td>
<td>63.57 76.96</td>
<td></td>
</tr>
</tbody>
</table>

Results on Paris6k and Oxford5k We consider this experiment primarily for comparison with progressive local filter pruning (PLFP) [74], which performs symmetric testing with a pruned version of VGG16. For the sake of comparison, there is no whitening in this case. According to Table 4, MobileNetV2 has substantially lower FLOPS and parameters than the pruned VGG16, yet with either teacher it performs better on Paris and nearly the same on Oxford.

Asymmetric embedding visualization Figure 3 visualizes the embeddings of a number of ROxford5k images, each obtained by a teacher and a student model. In asymmet-

dic testing, matching teacher and student features requires the same absolute coordinates in the feature space. This is done best by regression, but Contr+ also works well. Losses that rely on pairwise or higher-order relations, like RKD or DR, fail in this task. Triplet also fails, presumably because it also relies on relations.

5. Conclusions

There are certain unexpected or surprising findings in this work. First, regression is particularly effective in knowledge transfer. It appears that the more the constraints on student mappings (e.g. ranking $\rightarrow$ distance/angle relations $\rightarrow$ positions), the better the performance in standard symmetric testing. Second, the standard contrastive loss is particularly effective with asymmetric similarity at training, outperforming by a large margin state of the art methods like multi-similarity. A straightforward combination with regression—treating the anchor itself as positive—performs best on symmetric testing. These two solutions are the only ones where knowledge transfer helps, i.e., outperforms the student trained alone. In the new asymmetric testing task, regression is unsurprisingly a winner.

We have shown that using the original metric learning task while transferring knowledge is still beneficial in symmetric testing. It remains to be investigated whether the same can happen in asymmetric testing. We consider the same dataset in teacher and student training, so the latter is as supervised as the former. An interesting extension would be to consider a different, unlabeled dataset in student training. This would be a semi-supervised solution, like data distillation [55].
References


