Pattern Recognition 112 (2021) 107795

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Fast hard negative mining for deep metric learning

Bojana Gajić[a,b,c,*], Ariel Amato[a], Carlo Gatta[a]

[a] Vintra Inc, San Jose, CA, United States
[b] Computer Vision Center, Barcelona, Spain
[c] Universidad Autonoma de Barcelona, Barcelona, Spain

A R T I C L E   I N F O

Article history:
Received 5 November 2019
Revised 28 September 2020
Accepted 13 December 2020
Available online 17 December 2020

Keywords:
Deep metric learning
Instance retrieval
Re-identification
Siamese networks
Online hashing

A B S T R A C T

Deep metric learning methods aim to measure similarity of data points (e.g. images) by calculating their distance in a high dimensional embedding space. These methods are usually trained by optimizing a ranking loss function, which is designed to bring together samples from the same class while separating them from samples from all other classes. The most challenging part of these methods is the selection of samples that contribute to effective network training. In this paper we present Bag of Negatives (BoN), a fast hard negative mining method, that provides a set, triplet or pair of potentially relevant training samples. BoN is an efficient method that selects a bag of hard negatives based on a novel online hashing strategy. We show the superiority of BoN against state-of-the-art hard negative mining methods in terms of accuracy and training time over three large datasets.

1. Introduction

Metric learning, as a part of general machine learning, gained a lot of attention in the recent years [1–12]. The methods designed to embed images with similar content close to each other and separate them from images of other objects usually struggle when images are taken by different cameras, under different light conditions, from different angles, etc. Some of the recently published works propose solving metric learning problem by training deep neural networks that are computationally very expensive and task-specific [3–6,8,13–15]. Others require direct image-to-image comparisons, which results in methods that scale poorly at testing time [16,17]. Finally, one line of research focuses on simpler domain-agnostic architectures, that are able to adapt to any ranking problem [1,2,18,19].

Once the architecture is designed, the question that naturally arises is: what should be the appropriate loss function for training? Several works use classification loss [20,21]. These methods typically train a deep convolutional neural network for solving a classification problem, while extracting image descriptors by pooling features from the last convolutional layer at testing time. Even though these approaches are straightforward to implement, they struggle from serious limitations in terms of the number of classes that they can handle. For instance, a dataset with more than 10 M classes[1] (such as the FaceNet dataset, as reported in Kemelmacher-Shlizerman et al. [22]) and with a 128 embedding size, would require a model with more than 1 billion parameters, which would be impractical to train without severe overfitting.

Given the previously mentioned limitations of classification loss, alternative functions, widely known as ranking losses, have been proposed. The main goal of these losses is bringing the representations of samples from the same class close to each other, while separating the representations of images that belong to different classes. To do so, Siamese architectures with several parallel streams, which are made of the same networks with shared weights, are used [2,18,19]. These architectures allow simultaneous computing of compact representations for several input images, and combining them with a ranking loss. Even though these approaches do not depend on the number of classes in the training set, they are hard to train due to the complexity of finding input samples that are producing a loss greater than zero [2].

In this paper, we propose an online strategy for mining samples, that contributes to a more efficient training of Siamese architectures, while providing better validation scores on several datasets. We tested our method on datasets large enough so that the retrieval and re-identification problems cannot be easily solved using a classification loss. We use a large person re-ID dataset by merging publicly available datasets (similarly to [21]) and we show the

* Corresponding author at: Universidad Autonoma de Barcelona, Barcelona, Spain.
E-mail addresses: bgajic@cvc.uab.es (B. Gajić), aamato@vintra.io (A. Amato), cgatta@vintra.io (C. Gatta).

[1] In this paper we propose a general method which can be used for retrieval, re-identification or recognition, and therefore we treat identities or IDs and classes equally, and refer to them as *classes*.

https://doi.org/10.1016/j.patcog.2020.107795
0031-3203/© 2020 Elsevier Ltd. All rights reserved.

results on the publicly available retrieval datasets Stanford Online Products [7] and DeepFashion [23].

2. Related work

2.1. Hard negative mining

The problem of finding relevant candidates for ranking losses (especially for triplet loss) has received a lot of attention in the recent years for both retrieval [2,7,19,24–26] and tracking [27]. One research line bypasses this problem by proposing modifications of softmax loss for easier training [28,29].

A research line focuses on offline sampling approaches. An offline re-weighting of the loss can improve the quality of negative samples, but at non-negligible cost [24]. Taking advantage of extra knowledge on sub-categories within the dataset is also advantageous in mining negative samples [25].

Another group of methods, widely known as online hard negative mining (OLHN), take advantage of the samples representations available at mini-batch level in order to improve the probability of retrieving relevant negatives for the triplet loss [2,7,19,26]. Most of these works create mini-batches of kl images, k random images per each one of l random classes. The pioneer approach was introduced by Schroff et al. [2], called *semi hard* loss, where triplets are created by all anchor-positive pairs in a mini-batch. The negative sample is chosen so that the loss is in between 0 and the predefined margin α (see the Eq. (1)).

Lifted Embedding loss is proposed in Oh Song et al. [7], where the authors sample negative as the one closest to either anchor or positive for each anchor-positive pair in the mini-batch.

In [19] the authors propose two strategies for sampling inside of a mini-batch, which are extensions of the Lifted Embedding loss. *Batch all* loss is obtained by all possible combinations of triplets inside of batch. *Batch hard* loss takes all the images from the mini-batch as anchors of triplets. The positive is selected as the furthest sample from the same class as the anchor in the mini-batch, while the negative is the closest to the anchor from all the samples from different classes in the mini-batch.

Curriculum sampling is proposed in Wang et al. [26], where the beginning of training is performed using easy negative instances, and complexity increases through time. For each anchor they sort all negatives from the mini-batch according to their distances to the anchor, and they sample the negative with Gaussian distribution $\mathcal{N}(\mu, \sigma)$. μ and σ are changed through time, so that μ goes from max distance to min distance, and σ reduces.

All of these approaches have the same drawback: they focus on the local distribution of data inside of a mini-batch, while sampling the candidates for mini-batch randomly. A mini-batch created randomly is a good representation of the global distribution, but it does not represent the local embedding space. *As relevant negative samples could be found in the local neighborhood of the anchor sample, the probability of sampling useful triplets rises if the mini-batch is created from samples that belong to the same local neighborhood.*

Another research line comprises methods that use adversarial samples for metric learning [30,31]. In [30] the authors propose a way of training Siamese networks by generating adversarial, potentially hard, negative samples for training with variations of the triplet loss. The descriptors of all three input images are used for generating a synthetic, hard negative descriptor. This descriptor, together with the anchor and positive, forms a triplet of descriptors that is used for calculating the loss. Similarly, in Chen et al. [31] the authors propose a metric learning strategy that uses a set of real and a set of synthetic pairs for training.

The fourth research line proposes online strategies for providing relevant negative samples prior to mini-batch formation [18,32–34], and our contribution belongs to this research line.

In [32] authors propose a strategy that builds a tree of identities to facilitate the sampling of relevant negatives for a given anchor. The method clearly improves the quality of negative samples but at the cost of updating the tree at every epoch. Also, the tree construction is based on an identity-to-identity distance matrix, which thus scales quadratically with the number of identities.

In [18] authors explicitly face the problem of training a Siamese network with 100k identities. The basic idea is to generate a representation for each identity, and apply clustering on all the identities to generate clusters or subspaces, wherein identities are similar in each subspace. Authors propose to train a classifier on a subset of identities, then use the classifier to generate image representations, and finally perform k-means clustering in order to form the subspaces. Authors do not update the clustering during the training, thus the subspaces could become sub-optimal in later stages of the training.

In [33] authors propose a strategy in which anchor samples are compared to “class signatures” in order to limit the number of sample to sample comparisons. A stochastic process is added to avoid the oversampling of overly-difficult or noisy classes. The class signatures are constantly updated thanks to an additional classification loss. The method could be seen as a stochastic extension of [32] where the similarity between classes is computed at each step, while the class signatures are updated in a more efficient way. Nonetheless, the number of additional distance computation w.r.t. a pure random sampling is proportional to the number of classes.

In [34] authors propose a strategy for creating triplets of images from a subset of approximate nearest neighbors of the anchor image. This strategy requires a forward pass on all training set images at the beginning of each epoch, followed by graph construction, and search inside of the subspace. This strategy increases the speed w.r.t exhaustive search (“…*Given that $\mathcal{O}(N^2)$ is the best case complexity for the naive hard mining approach above, we can conclude that our method is computationally more efficient*…”[34]), while providing relevant triplets. Similarly to [32], the main issue of this approach is that triplets are formed at the beginning of each epoch. This strategy for triplet formation can be useful for small datasets, but does not guarantee high quality of sampled triplets towards the end of the epoch when trained on datasets with large number of images.

The main drawbacks of the approaches from [18,32–34] are: (1) high computational cost and (2) they scale poorly with the number of classes.

2.2. Online hashing methods for image retrieval

In [35] authors design a hashing method for cross-domain retrieval. They split the full image or text representation into two parts: one that is domain specific, and the second that is shared for the two domains. The two parts are in the end concatenated, and used as unique image/text representations.

In [36] authors propose a method that learns binary codes for subspaces, based on a similarity preserving criterion. This method uses the binary codes to train binary classifiers which are further used as hashing functions.

In [37] authors propose an unsupervised way to train a hashing method for image retrieval. They propose a loss function that penalizes the network if two similar images are put into different hash bins, as well as if two dissimilar images are assigned to the same hash bin. They measure image similarity based on distances of image descriptors extracted from ImageNet pre-trained ResNet152 network.

The same loss for assigning images to the hash bins is optimized in Deng et al. [38]. This method has an additional branch that assigns class labels to the same hashes as the images that be-

Table 1
Comparison of sampling strategies.

Strategy	n_e	n_d	n_t	Additional computation
Random	$\mathbf{0}$	$\mathbf{0}$	$m/3$	–
Semi-hard	0	$(2b^2 - 2b)l$	$m/3$	–
Batch hard	0	$(9b^2 - 2b)l$	m	–
Exhaustive	$l(N-b)$	lN^2	$m/3$	–
Hierarchical Tree [32]	N	$N^2/2$	N.A.	pre-training
Smart mining [34]	N	$(N/i)^2$	N.A.	extra global loss
100k IDs [18]	0	0	N.A.	classifier, all feature extract, k-means
Stochastic class-based [33]	$k(K-1)$	$k\left(\frac{N}{n} + (K-1)\right)$	N.A.	classifier, class signatures
Spectral Hashing [39]	N	0	N.A.	PCA
Bag of Negatives	$\mathbf{0}$	$\mathbf{0}$	N.A.	autoencoder

long to that class. As this is a supervised hashing method, a pair of images is considered to be similar if both of them belong to the same class, and dissimilar otherwise. Even though this method performs well on image retrieval task, it is not designed for hard negative mining; if the number of hashes is slightly greater than the number of classes in the train set, each class can potentially be assigned to one hash; using this table, sampling images for a mini-batch from one hash bin would not provide any negatives, since all the samples would belong to the same class. If the number of hashes is smaller than the number of classes, the system will be unstable.

On the other hand, unsupervised hashing proposed in Deng et al. [37] is suitable for hard negative mining task. We show results and discuss its advantages and disadvantages in Sections 3, 6.3.2 and 7.

3. Motivation

The triplet loss (see Eq. (1)) is based on the construction of triplets $i \in \mathcal{T}$ formed by an anchor sample x_i^a, a positive sample x_i^p (belonging to the same class as the anchor) and a negative sample x_i^n. The samples are mapped into an embedding by a given function $f(\cdot)$, that is usually a deep convolutional network, which parameters are learned by means of minimization of the loss $\mathcal{L}$.

$$\mathcal{L} = \frac{1}{n_t} \sum_{i \in \mathcal{T}} \left[||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha \right]_+ \tag{1}$$

The goal of the triplet loss is to ensure that the anchor-negative pairs are far from each other by a margin α with respect to the anchor-positive pair distance. It is well known that the most challenging part of using the triplet loss to train a metric learning system is generating triplets that produce a non-zero loss. This is hard, since the number of all possible triplets in the dataset is proportional to the cube of total number of images N in the dataset, $|\mathcal{T}| \sim N^3$, and the more the system trains, the less probable it is to find a negative for a given anchor-positive pair that provides a non-zero loss [2].

Let n be the average number of images per class, m the mini-batch size, k the number of images of each class in the mini-batch, and l number of steps per epoch. For the sake of clarity, we introduce the notation $\hat{n}$, the number of negative samples that produce a non-zero loss if used in conjunction with the triplet loss and an anchor-positive pair. The more we train the Siamese network the smaller $\hat{n}$ becomes.

We propose a systematic cost analysis given a sampling method in terms of n_e, the extra number of forward passes to be computed per epoch, and n_d, the extra number of distances to be computed in order to select a set of negatives per-mini-batch over an entire epoch. Additionally, we report the number of triplets per mini-batch n_t, summarised in Table 1.

The "quality" of the retrieved negatives is also relevant, as pointed out in Wu et al. [24]: negative samples have to be distributed such that the anchor-negative distance is almost uniformly distributed. More on the latter topic will be discussed in the experimental section.

Sampling the negatives randomly from the whole dataset has complexity $\mathcal{O}(1)$ but does not provide relevant negative samples except at the beginning of the training, since $p_{\hat{n}} = \hat{n}/(N-n) \simeq \hat{n}/N$. From now on we will omit n from the formula since it is negligible w.r.t. N.

Semi hard loss [2] employs a negative sampling strategy that has an increased cost due to the fact that the additional computed distances scale polynomially with the mini-batch size. The improvement in $p_{\hat{n}}$ with respect to the random sampling is linearly dependent to the number of triplets b. For this reason, authors use huge mini-batches in the order of 1800 samples. $p_{\hat{n}}$ is thus increased to $2b\hat{n}/N$, at the cost of large mini-batches and additional computation.

Batch hard loss [19] is an improved version of the *semi hard* loss where, thanks to a more controlled mini-batch creation and additional distances computation, the method exhibits $p_{\hat{n}} = m\hat{n}/N$. This strategy offers a 50% improvement in $p_{\hat{n}}$ w.r.t. the *semi hard* approach, but still provides a probability that depends on the mini-batch size. The additional cost in the distances computation is mitigated by a 3 times factor in the numbers of computed triplets.

An offline exhaustive search into the dataset provides $p_{\hat{n}}$ =min$(3\hat{n}/m, 1)$. This is, of course, not viable for large datasets. Nonetheless, for relatively small datasets and with the proper sampling strategy over the $m(N-m)$ distances, exhaustive search provides excellent negatives samples [24].

Hierarchical Tree sampling [32], 100k IDs [18], Smart Mining [34] and Stochastic class-based hard example mining [33] are methods for sampling candidates prior to mini-batch creation. Those methods can be combined with online hard mining strategies (such as semi-hard and *batch hard*) and further increase the probability of sampling relevant negative samples.

In [32] authors propose sampling identities based on inter-class distances. The main drawback of this method is the high computational cost of creation of the inter-class distance matrix. This matrix should be updated once per epoch, and it requires forward passes of the whole dataset ($\mathcal{O}(N)$), and calculating all-vs-all sample distances ($\mathcal{O}(N^2)$).

The method proposed in Wang et al. [18] for batch generation is based on hashing. This approach is faster than Hierarchical Tree, as it does not require any additional distance calculations nor extra embedding extraction. Its drawback is the complexity of generating the hash table, as it requires training a classifier on a subset of the dataset, extracting of features of all images from the train set and executing k-means clustering.

Table 2
Time required for training for 100k steps and until convergence.

Method	Time for 100k steps [h]	Convergence time [h]
batch hard	12.3	34.4
BoN-*batch hard*	12.6	10.1

Table 3
mAP validation results at peak performance for every method.

Method	#steps	Market		Duke	
		map	r1	map	r1
Random	600k	28.1	47.5	22.5	37.6
BoN-Random	440k	61.4	80.3	51.3	70.2
semi hard	350k	59.3	76.8	53.5	71.3
batch hard	280k	60.8	78.6	53.7	70.6
batch hard - contrastive	120k	48.9	66.9	37.8	56.5
SPL (reproduced) [37]-*batch hard*	200k	65.3	81.5	59.3	75.8
HT(reproduced) [32]-*batch hard*	310k	65.9	82.8	57.5	74.9
100k (reproduced) [18]-*batch hard*	90k	67.8	83.3	61.2	77.7
BoN-*batch hard* - contrastive	90k	59.4	77.0	51.9	70.6
BoN-*batch hard*	**80k**	**69.5**	**85.2**	**62.1**	**78.5**
SH-*batch hard*	100k	71.6	86.6	62.9	78.2
batch hard (2x batch)	70k	62.9	80.3	56.7	74.7

The method called Smart Mining [34] uses samples from approximate nearest neighborhood to create potentially relevant triplets. However, in the beginning of each epoch one full forward pass of the whole dataset is performed. In addition to this, in each training step $(N/i)^2$ distances are computed, where i is the number of neighborhoods.

Stochastic class-based hard example mining [33] is a method that uses class signatures when creating triplets. This approach requires $k(K-1)$ extra forward passes in each training step and kN/n distances, where K is the number of classes in mini-batch.

This brief study shows that the efficiency of relevant negative mining is a crucial issue. Also, increasing the probability of picking a relevant negative is key to the improved performance from *semi hard* to *batch hard* strategy. Scalability to very large datasets with a large number of classes is a necessity within the training of Siamese architectures.

In this paper we propose a novel method for batch creation, inspired by Spectral Hashing [39]. In contrast to Spectral Hashing, which requires additional forward passes of all images from the dataset, our method updates the hash table online, with negligible computational cost.

4. Contribution

The paper's *main contribution* is a computationally inexpensive and mini-batch size independent, online strategy for improved negative mining in large datasets; which we named Bag of Negatives (BoN). The main advantages of BoN w.r.t. previous methods are: (1) fewer training steps to converge, (2) a better performance on validation sets due better sampling of negatives, and (3) a negligible additional computational cost w.r.t. the Siamese architecture training.

We stress that our methodology does not require computing additional samples representation nor their respective distances to be able to select appropriate negatives (see Table 1). It can be combined with any loss that requires a negative sample, as shown in Sections 5.4 and 5.5. Nonetheless, for simplicity, we will analyse the behaviour of BoN with triplet based losses since they perform better than contrastive loss (Table 3); analogous results are achieved with other losses, such as quadruplet loss. We also want to emphasize that this approach has been devised with large datasets and computational efficiency in mind. Finally, the method has only one relevant meta-parameter, discussed in Section 6.

5. Bag of Negatives

A negative sample whose representation is close to the anchor sample provides a triplet that is more likely to produce non-zero loss. The main purpose of BoN is providing these relevant negative samples using an algorithm that is computationally inexpensive.

BoN is inspired by the Spectral Hashing method [39]. Nonetheless, we had to introduce several changes in order to efficiently adapt it to the negative mining problem during training.

Spectral Hashing is a nearest neighbour search algorithm that is shown to perform better than Product Quantization while being simpler to implement and more efficient at learning the hash function [39]. In terms of performance, it is inferior with respect to methods that address the embedding compression and the quantization as a whole problem, e.g. [40,41]. However, we have to consider that the embedding is changing during the training, thus a simpler but flexible approach is preferred over methods providing better results at a greater computational cost.

The main approach of the Spectral Hashing method is to (1) learn a linear projection from the embedding space (of size e) to a lower dimensional space (of size $s \ll e$) by means of a standard PCA, (2) apply the projection to a sample, (3) perform a 1 bit quantization over every dimension by threshold at 0, and (4) group the s bits into an integer codeword (line 4 in Algorithm 2). The codeword represents the entry of a hash table. The underlying assumption is that samples falling into the same bin are neighbours in the high dimensional embedding. Of course, this assumption is over-optimistic and the variation from the optimal are mainly due to the following facts: (1) being $s \ll e$ we lose some information about the topology of the high dimensional embedding and (2) the quantization is harsh and there is no actual control on the quantization error during the process. Nonetheless, experimental validation shows that the Spectral Hashing method is indeed performing well in retrieval tasks [39].

However, the direct application of the Spectral Hashing (or any other nearest neighbour algorithm) method to the problem of retrieving negative samples is not straightforward because the embedding is dynamically changing during the training. One can compute the whole embedding every certain number of steps, plus computing the PCA, and the hash table, but this naïve strategy does not scale well for large datasets. Consequently, we propose three main modifications to the Spectral Hashing approach in order to have an **online** algorithm that mimics its performance (see Fig. 1 and Algorithm 1):

Algorithm 1 Main algorithm.

```
1:  Input: $image\_names$, $image\_labels$, $s$, $batch\_size$, $n\_ids$
2:  $hash\_table, C, \mu \leftarrow hash\_init(image\_names, image\_labels, s)$
3:  $net \leftarrow$ ImageNet_init
4:  $autoencoder \leftarrow$ random_init
5:  while not converged do
6:      $images, labels \leftarrow create\_mini\_batch(image\_labels, hash\_table,$
        $batch\_size, n\_ids)$
7:      $descriptors \leftarrow net(images)$
8:      $hash\_vectors \leftarrow autoencoder(descriptors)$
9:      $hash\_table, C, \mu \leftarrow hash\_table\_update(hash\_table, C, images,$
        $labels, hash\_vectors, \mu)$
10:     backpropagate
11: end while
```

1. The PCA is substituted by a linear auto-encoder paired with L_2 reconstruction loss (Algorithm 1 line 8).

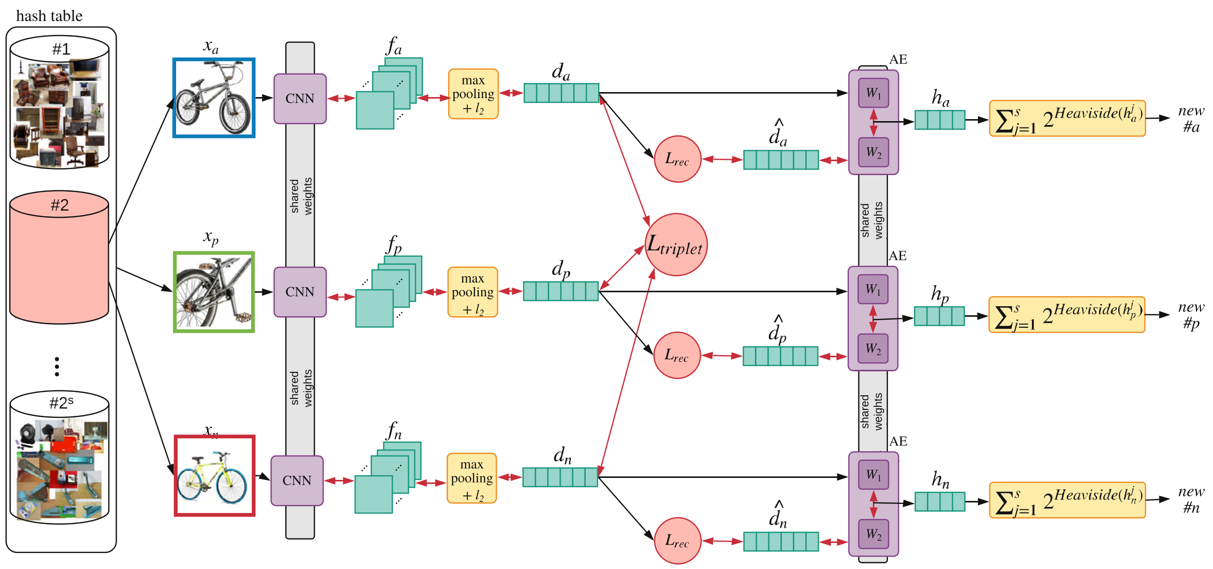

Fig. 1. BoN strategy. Triplets with good quality negatives are formed using the information from the hash table. The resulting embedding is used to learn both the deep model and a linear projection that, in turn, provides a low-dimensional embedding. Its quantization provides (possibly) new entry positions in the hash table for the input images. The hash table and the linear autoencoder are updated at each training step with minimal overhead.

2. The quantization threshold is dynamically estimated, per-dimension, instead of being fixed to 0 (Algorithm 2 line 9).
3. The hash table is dynamically updated (Algorithm 1 line 9).

Algorithm 2 Hash table update.

```
1:  function HASH_UPDATE(hash_table, C, image_indexes, image_labels, hash_vectors, μ)
2:      for (image_idx, image_label, hash_vector) in (image_indexes, image_labels, hash_vectors) do
3:          old_hash ← C[image_idx]
4:          hash ← $\sum_{j=1}^{s} 2^{\text{Heavyside}(hash\_vector[j]-\mu[j])}$
5:          if hash ≠ old_hash then
6:              remove (image_idx, image_label) from hash_table[old_hash]
7:              add (image_idx, image_label) to hash_table[hash]
8:              C[image_idx] ← hash
9:              $\mu \leftarrow \beta\mu + (1-\beta)\mu$
10:         end if
11:     end forreturn hash_table, C, μ
12: end function
```

5.1. Linear auto-encoder

Since the **online** PCA estimation is in general computationally inefficient and potentially numerically unstable [42], we train a linear autoencoder (AE) paired with L_2 reconstruction loss, as in formulas (2), where $\boldsymbol{h}(\boldsymbol{x})$ is the projected sub-space of dimensionality s. The reconstruction loss should not modify the embedding space, therefore the gradients generated by the $\mathcal{L}_{AE}$ loss are back-propagated only through the fully connected layers of the autoencoder.

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{x}) &= \boldsymbol{W_1 f}(\boldsymbol{x}) + \boldsymbol{b_1} \\ \hat{\boldsymbol{f}}(\boldsymbol{x}) &= \boldsymbol{W_2 h}(\boldsymbol{x}) + \boldsymbol{b_2} \\ \mathcal{L}_{AE} &= ||\boldsymbol{f}(\boldsymbol{x}) - \hat{\boldsymbol{f}}(\boldsymbol{x})||_2^2 \end{aligned} \quad (2)$$

This approach can approximate a PCA computation, but it also allows non-orthogonal representations. The AE continuously models the projection that provides the codeword to the hash table update procedure. The added cost of learning such AE is negligible w.r.t. the Siamese network training. The choice of s is related to two factors: (1) the smaller s, the more difficult to reconstruct (in the L_2 sense) the original sized embedding and (2) the bigger s, the larger the number of bins obtained after the binarization, more precisely $B = 2^s$. A detailed analysis on the behaviour of BoN as a function of s is presented in the experimental section.

5.2. Dynamic quantization thresholds

Since the lower dimensional space $h(x)$ changes dynamically during the training and the AE does not guarantee a zero mean hidden representation, the correct thresholds μ for its binarization have to be estimated as a running mean: $\mu \leftarrow \beta\ \mu + (1-\beta)\ h(x)$, where β controls how quickly the running average forgets old samples (Algorithm 2 line 9). In our experiments we noticed that varying $\beta \in [0.95\,0.999]$ does not influence the results so that it is not a critical parameter to tune.

5.3. Hash table dynamic update

We maintain an hash table L (*hash_table* in Algorithm 2) that, for each entry indexed by an integer j, contains a collection of images identified by pairs $(v, I(v))$, where v is an integer that uniquely represents an image in the dataset (*image_idx* in Algorithm 2), and $I(v)$ is an integer that uniquely represents the class of the given image (*image_label* in Algorithm 2). Also, we keep track of the latest hash entry for every image in the dataset, using integer values, such that $C[v] = j$ (line 8 in Algorithm 2). In such a way, updating the hash table has a very limited computational cost. The slowest part of the update procedure is removing the tuple $(v, I(v))$ from the bin to which it had been assigned (line 6 in Algorithm 2), and it has a cost of $\mathcal{O}(N/2^s)$. In term of memory cost, assuming that both the classes $I(v)$ and the sample v identifiers can be represented with 4 bytes integers, we need only a total of $4(N+2N)$ bytes to store both the hash table L and the hash entry C. As an example, even a very large dataset with 10M images requires only 115 Megabytes for the hash table. The update procedure is described in Algorithm 2.

5.4. Bag of Negatives with triplet loss

The simplest way of using BoN is to create mini-batches by randomly sampling b anchor-positive image pairs. For each pair, we sample a negative image randomly among the images that belong to the same bin as the anchor. In case the anchor belongs to a bin in which there are no other images from a different class, we sample the negative image randomly from the whole dataset.

5.5. Bag of Negatives with batch hard loss

As BoN is able to provide relevant images for batch sampling, it can be easily combined with a loss such as *batch hard*. It is important to create batches of size m, which contain k images from l classes for *batch hard* (see Algorithm 3). We set $k = 2$ for all the experiments, as we are focusing on showing the importance of good negative sampling, and we want to avoid the results being influenced by hard positive sampling. We randomly sample l classes that belong to the same bin as the first, random sample (lines 4–9 in Algorithm 3). If the bin has only one element, we sample the rest of the images needed for the batch randomly (line 10 in Algorithm 3). In case the number of classes in the bin is greater than one and lower than l, we append the missing classes from another bin, which is randomly chosen. The process is repeated until sampling l classes. Once we have a set of l classes, we choose k images randomly from the images that belong to that class (lines 16 and 17 in Algorithm 3).

Algorithm 3 Mini batch creation.

```
1: function CREATE_MINI_BATCH( $image\_labels$, $hash\_table$, $batch\_size$, $n\_classes$)
2:     $batch\_labels \leftarrow$ empty list
3:     while length($batch\_labels$) $<$ $n\_classes$ do
4:         $anchor\_idx \leftarrow$ random((1 to length($image\_labels$)), size=1)
5:         $anchor\_hash \leftarrow C[anchor\_idx]$
6:         if $anchor\_hash > 0$ then
7:             $classes \leftarrow$ classes from $hash\_table[anchor\_hash]$
8:             $new\_classes \leftarrow$ random($classes$, size = min(length($classes$), $n\_classes -$ length($batch\_labels$))
9:         else
10:            $new\_classes \leftarrow$ random((1 to max($image\_labels$)), size= $n\_classes -$length($batch\_labels$))
11:        end if
12:        add $new\_classes$ to $batch\_labels$
13:    end while
14:    $batch\_idxs \leftarrow$ empty list
15:    for $class$ in $batch\_labels$ do
16:        $class\_idxs \leftarrow$ indexes of $class$ in $image\_labels$
17:        add random($class\_idxs$, size=$batch\_size/n\_classes$) to $batch\_idxs$
18:    end forreturn $batch\_idxs$, $batch\_labels$
19: end function
```

6. Empirical evidence

6.1. Datasets

Person re-identification large dataset. We merged eleven publicly available datasets for person re-identification, CUHK01 [43], CUHK02 [44], 3DPeS [45], VIPeR [46], airport [47], MSMT17 [48], Market-1501 [49], DukeMTMC [50]. The merged dataset has 10.5k IDs, and 178k images. We used both training and testing partitions of all the datasets except for Market-1501 and DukeMTMC-reID and we did not use the images that are labeled as distructors or junk.

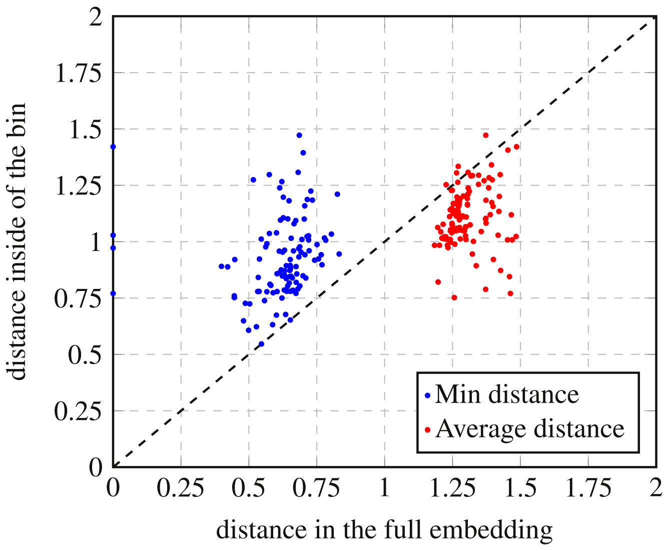

Fig. 2. Negative distances calculated in the whole dataset (x-axis) vs. negative distances calculated inside of bins (y-axis) for 100 anchors.

Stanford online products Oh Song et al. [7] is a retrieval dataset which contains 120k images of 22.6k products. The dataset is split into two partitions, the training one, which contains 59.5k images of 11,3k products, and testing, 60.5k images of 11.3k classes.

DeepFashion - in-shop clothes retrieval Liu et al. [23] is a part of DeepFashion dataset which is designed for instance retrieval. The dataset is made of 54.6k images of 11.7k clothing items. All the images are taken under controlled conditions.

6.2. Pre-trained backbone and training parameters

We use Inception-V3 as a backbone for our model. In particular, we take the convolutional layers and initialize them with weights from a standard network pre-trained on ImageNet. The final descriptors are further globally max-pooled and ℓ_2 normalized. The descriptors size is 2,048. The model is trained using ADAM optimizer, with the initial learning rate 10^{-4}, and with learning rate decay 0.9 each 50k iterations. The images for person re-ID are resized to 192×384 pixels. At test time, we extract representations and compare them using the dot product.

6.3. Analysis of Bag of Negatives

In this section we answer the following relevant questions: (1) How does BoN compare to the exhaustive search? (2) How does BoN behave in terms of non-zero loss triplets? (3) How does BoN perform changing the subspace dimension s? (4) How much overhead it adds to the training? and (5) How stable is the hash table during training? We provide all the analysis on the person re-identification dataset, as it is challenging, as well as appropriate for testing of all the algorithms mentioned in the Section 3.

6.3.1. BoN vs. exhaustive search

In the motivation section we explained that an exhaustive search in the embedding provides always a non-zero loss negative sample, if existing. Nonetheless, its cost is prohibitive when dealing with large datasets. Fig. 2 shows the distance between 100 anchor samples and a set of negative samples at 200k steps of training for BoN-random with $s = 18$. The abscissa shows the average distance (red) and the minimal distance (blue) when applying an exhaustive search over the whole training partition of the large person re-ID dataset. For the very same anchor sample, the ordinate shows the

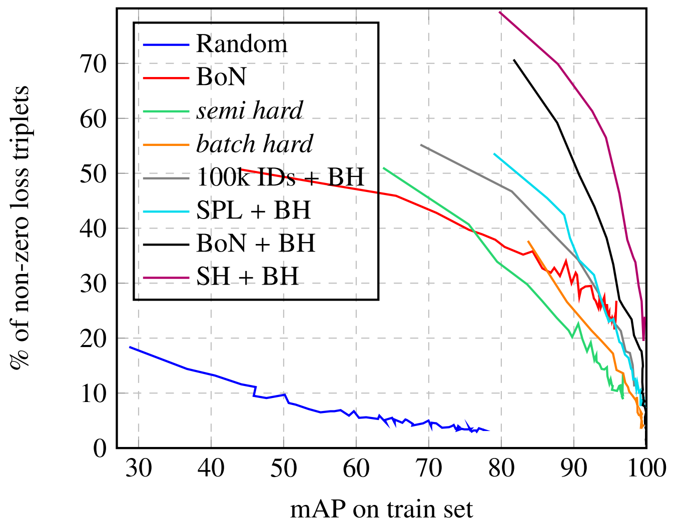

Fig. 3. Percentage of non-zero loss triplets per mini-batch as a function of mAP on the training set.

distance where the search is limited to the samples belonging in the same hash bin as the anchor.

An ideal hashing method would exhibit blue dots on the diagonal of the plot, meaning that every hardest negative sample (closest to the anchor) lies in the corresponding anchor bin. As it can be seen, the blue dots are not deviating excessively from the diagonal, which means that BoN is able to retrieve negatives of good quality. Again, for the purpose of training a Siamese architecture with triplet loss, using always the hardest negative can be disruptive and particularly dangerous due to mislabeled samples [24]. This is particularly true and the authors of [33] introduce stochasticity in order to avoid this problem.

The analysis of the red dots is also of interest: as expected, on average, sampling from the reference bin provides samples that are closer to the anchor sample w.r.t. random sampling from the whole dataset.

It is worth mentioning that the distance between the minimal and average distances in the full embedding space is 0.7. This means that the data distribution is sparse and that the random sampling can lead to choosing negative samples which are far from being hard. On the other hand, the distribution of the distances inside of a bin has smaller standard deviation, as the distance between the minimal and average distances is 0.2. This allows us to use random sampling inside of the bins, without decreasing the quality of the chosen samples.

6.3.2. Non-zero loss triplets analysis

Fig. 3 shows the percentage of non-zero loss triplets (measured at train time) as a function of the training mean Average Precision (mAP) for the random sampling, BoN-Random, the *semi hard* loss, the *batch hard* loss and BoN-*batch hard*, Spectral Hashing-*batch hard*, 100k IDs-*batch hard* and SPL-*batch hard* loss on the person re-ID dataset. For all methods the margin is set to $\alpha = 0.3$, the mini-batch size is $m = 48$, and the leftmost point on the plot for every method is obtained at 10k steps of training. As expected, the percentage of non-zero triplets for the random sampling (blue line) starts at only 20% and decreases as the mAP increases; at mAP $= 77.4$ the non-zero triplets are less than 5% and the training is virtually unable to learn anything else.

BoN-random (red line) significantly increases the number of non-zero loss triplets w.r.t. the pure random sampling, without modifying the loss nor the way the anchor-positive pairs are formed. The improvement is solely due to the improved sampling of negatives.

BoN-random exhibits a behaviour similar to *semi hard* (green line) and *batch hard* (orange line) while providing, in general, more non-zero losses triplets. However, the nature of the improvement provided by our method and *batch hard* is very different: BoN searches for negatives in a local region of the embedding space while the *batch hard* forms the triplets seeking for the non-zero triplets in an explicit way within a mini-batch but sampling from the whole embedding.

These two complementary strategies can be easily combined as seen in Section 5.5. The combination inherits the benefits of both approaches: at 10k steps *batch hard* and BoN-*batch hard* have a similar mAP $\approx 83\%$ but BoN-*batch hard* (black line) has about 2 times more non-zero loss triplets, and it has more non-zero loss triplets systematically until the end of the training. As it will be seen in the comparison, this behaviour not only speeds-up the training, but also provides better triplets, which leads to significant improvement of the performance on validation sets.

We measure the limitations of BoN by comparing it to the "gold standard", Spectral Hashing. The combination of Spectral Hashing and *batch hard* requires the following steps: (1) feature extraction on the whole training set, (2) reduction of the feature size by PCA to the size s ($s = 18$) and (3) hash table construction; we repeat this procedure every 5k steps. Given this hash table, batches are created the same way as explained in Section 5.5. BoN-*batch hard* shows very similar behavior to the Spectral Hashing - *batch hard* (magenta line): they both train quickly, obtaining almost the same mAP after 10k steps, with high percentage of non-zero loss triplets. During the whole training Spectral Hashing - *batch hard* is providing more non-zero loss triplets. This is expected, as the hash table is updated at the same moment for all the samples. However, this configuration does not scale for datasets with large number of images.

We analyze the behavior of batch creation proposed in Wang et al. [18], using 10 clusters as suggested by the authors. We use these clusters for creating the hash table and we do not update it during the training. In addition to longer training time, this method lacks flexibility in updating the hash table. In other words, samples that are considered relevant negatives to an identity are set at the beginning of the training and are static w.r.t. the training process. Moreover, a possible sub-optimal clustering is going to be seriously detrimental to the training. In the beginning of the training, this method obtained lower mAP on the train set (gray line) while having more non-zero loss triplets than *batch hard*. The number of relevant triplets in the end of the training decreases, and both accuracy and the percentage of the non-zero loss triplets are inferior to BoN-*batch hard*.

Even though Semantic-Preserving Loss (SPL) [37] has not been designed as a hashing method for hard negative mining, we consider it relevant to our work, and thus we adapted it to this purpose. We use SPL loss (Eq. (3)) as a replacement of reconstruction loss in BoN. In this case, the encoder is a fully connected layer with tanh activation function, that maps image descriptors d_i into corresponding hash entries $h(x_i)$. Following the rationale proposed in Deng et al. [37], the similarity matrix S is a non-linear function of the dot product between images' descriptors within a mini-batch (4): a pair of similar images (with dot product above a certain *threshold*, set to 0.6) are mapped to 1, otherwise they are mapped to -1. The minimization of Eq. (3) should encourage the mapping of similar images to the same hash entry, thus providing useful

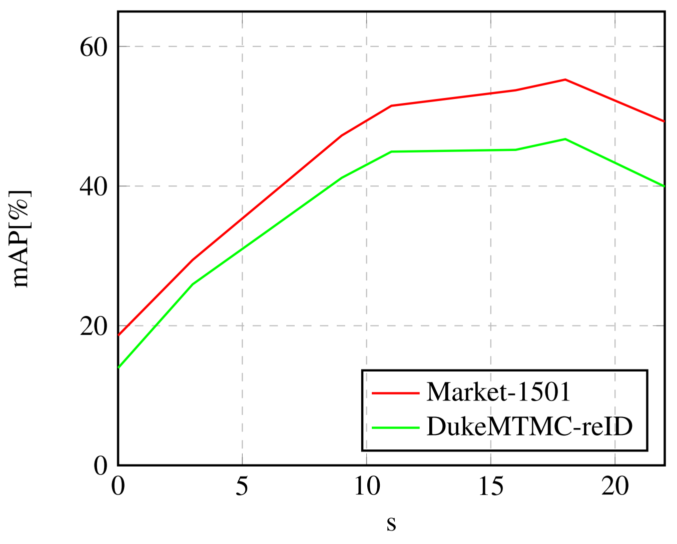

Fig. 4. Validation mAP as a function of s.

negatives samples as efficiently as BoN.

$$\mathcal{L}_{SP} = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\left(\frac{1}{s}h(x_i)h(x_j) - S_{ij}\right)^2 \tag{3}$$

$$S_{ij} = \begin{cases} 1, & d_i \cdot d_j > \text{threshold} \\ -1, & \text{otherwise} \end{cases} \tag{4}$$

BoN-BH shows superior results: it trains faster with more non-zero loss triplets (see Table 3). We believe that the advantage of BoN over SPL resides in the fact that the BoN AE loss does not depend on the relationship between mini-batch samples, thus provide a more stable hash table. Also, the quality of SPL mapping depends on the quality of the mini-batch sampling, which in turn depends on the hash table itself; such dependence can introduce a non-negligible instability in the training. Finally, in this context, SPL can be improved by adding an extra dedicated network that provides the mapping between the input *image* and the hash entry, instead of using the descriptor as an approximation of the input image; such a strategy could importantly increase the computational cost of the approach, and it is currently out of the scope of the paper.

6.3.3. *BoN-random behaviour varying s*

Being s the only relevant meta-parameter of BoN, we find extremely important to discuss its influence on BoN performance. It is interesting to note that BoN-Random degenerates to pure random sampling for $s = 0$. Fig. 4 shows the mAP results on the Market and Duke validation datasets for different values of s at 200k steps. As it can be seen, the performance increases with s and it reaches a maximum at $s = 18$; nonetheless, with $s = 22$, BoN reaches its breaking point and the average number of samples per bins (for non empty bins) is very low, such that BoN-Random starts to perform negative sampling in the whole dataset too frequently.

6.3.4. *Training time*

Table 2 presents the time needed for training a model for 100k steps, and total time needed for convergence for *batch hard* and BoN - *batch hard* methods, as BoN provides the best results when combined with *batch hard*. Both experiments are conducted under the same conditions; we trained the models on a TITAN X GPU with non-augmented images of size 384x192 pixels, using inception_v3 as backbone architecture, initialized with the weights obtained from ImageNet pretraining. The relative overhead that

Table 4
validation results at peak performance for every method and dataset. * stands for the best number found in literature that uses additional attention ensembles. ★ means that the method uses bilinear pooling.

Method	Stanford		inShop	
	r1	r10	r1	r10
lifted structured [7]	61.5	80.0	–	–
DAML [30]	68.4	83.5	–	–
hierarchical tree [32]	74.8	88.3	80.9	94.3
sampling matters [24]	72.7	86.2	–	–
ABE-8^{512} [52]*	76.3	86.4	87.3	96.7
Stochastic class-based [33]★	77.6	89.1	**91.9**	**98.0**
BoN-*batch hard*	**80.2**	**91.4**	91.4	97.9

BoN introduces is 3%. However, the model trained with BoN needs fewer steps to train, which means that total train time is reduced 3.4 times. In other words, BoN saves 24.26 h when trained with *batch hard* loss, while significantly improving the performance of *batch-hard*, as it will be shown in Section 7.

We additionally measured the time needed for one full forward pass of all the images in the train partition of the person re-identification dataset, which is independent on the sampling strategy, or loss function. The time to extract all the features is 11.5 min, which is equal to 1527 training steps of BoN-BH or 1572 steps of *batch hard*. All methods that require the computation of features in each epoch ([18,32–34] and Spectral Hashing) introduce an overhead of at least 42% at train time. BoN has equal or better performance than [18,32–34] (see Table 4) while adding one order of magnitude less overhead.

6.3.5. *Bin stability analysis*

The majority of recently published strategies for hard triplet mining take a snapshot of the full embedding in each epoch and create batches of hard triplets based on that information. In contrast, we create mini-batches based on an online hash table that stores all the training images in bins. In every training step, we move all samples from the mini-batch from old to new hash bins, which are approximated by the latent representations of the autoencoder that is trained to reconstruct the embedding. Even though this strategy introduces minimal computational overhead, it uses a noisy embedding approximation for triplet mining. In this section we analyse the tendency of images moving from one bin to another, as well as the Hamming distance between the former and current hashes (see Fig. 5).

In each training step we assign all $2l$ images (2 images from each of l classes) from the mini-batch to new bins. In the beginning of the training we sample one image per identity randomly from the list of images that have not been sampled as the first images, and the second image randomly. The images are initially not assigned to any bin, so all of them are added to the hash table as new (orange bar in Fig. 5). As the training continues, the number of newly inserted images is reducing, as both images per ID could have been sampled earlier in the training. While the hash table is not fully populated, the associated hash entry is unstable, and the images are moving from one bin to another frequently. Once the embedding becomes more stable, the percentage of images staying in the same bin increases significantly (green bar). However, around 90% of images still keep moving.

Fig. 5 shows Hamming distance between the old and new hash entries for all images processed in a mini-batch. It can be noticed that, after 25,000 steps, the bins become stable, and only 8% of images move to a bin that is on Hamming distance greater than 4. We set $s = 16$, which means that more than 75% of bits are kept the same. 92% of images are moving inside of the neighborhood of

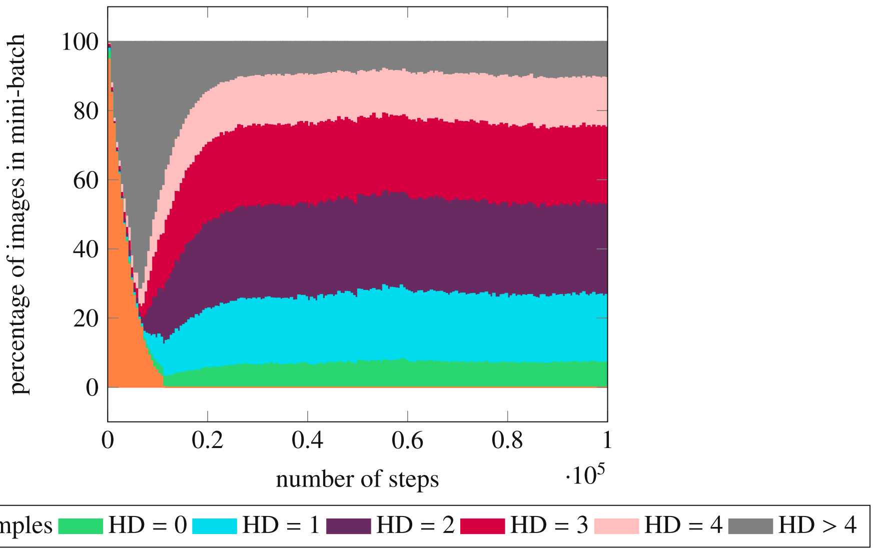

Fig. 5. The percentage of samples that were added to the hash table or moved from one bin to another. HD stands for Hamming distance between the old and new hash entry.

bins on Hamming distance smaller than 5 means that most of the images are moved to another bin with similar content.

The fact that images are moving from one bin to another is expected, and there are a couple of reasons for that. First, the decision boundary that separates bins is updated during training, which means that the samples that are close to the boundary can easily move from one bin to another. Second, the embedding changes through time, as does its compressed approximation, so an image that was assigned to one bin can move and be closer to some other samples in a different step of training. As mentioned above, the fact that images move in neighboring bins is not a problem; it is actually beneficial to avoid sampling negative samples that are either noisy or overly-difficult.

7. Results and comparison

In this section we perform a controlled comparison of our proposal with some of the most commonly used ranking losses: triplet, *semi hard* and *batch hard*, contrastive-*batch hard* and the three methods for triplet selection: hierarchical tree [32], 100k IDs [18] and SPL [37]. We avoid extra variables (e.g. augmentation, other architectures, etc.) that could mask the empirical results for other reasons not related to negative sampling and triplets construction. For such reasons, we use the same mini-batch size for all the methods, the same pre-trained back-bone, the same margin α and the same embedding size (see Section 6.2 for the details). Harwood et al. [34] and Suh et al. [33] are not included in this comparison, since they require an extra loss which can corrupt the analysis; a performance comparison with these approaches is provided in Table 4.

Table 3 shows the results of the comparison on the person re-identification dataset. As it can be noticed, BoN-random clearly outperforms pure random sampling in fewer steps and provides validation mAPs comparable to *semi hard* and *batch hard*. Even though BoN improves results of *batch hard* sampling when combined with the contrastive loss, it performs significantly worse than the original *batch hard* (combined with triplet loss), so we performed all the experiments using the *batch hard* - triplet loss setting. Spectral Hashing - *batch hard* outperforms BoN-*batch hard*, which is expected, considering that BoN is an online approximation of Spectral Hashing. The numbers show that the margin between BoN and Spectral Hashing is only 1.5% on average on the two evaluation datasets. However, Spectral Hashing can be used only if the train set is reasonably small; thus its application on bigger datasets would be unfeasible.

One can argue that the performance of BoN can be easily reached by just increasing the mini batch size of the *batch hard* method. The experiment *batch hard* (2x batch) in Table 3 shows a training in which the mini-batch size has been doubled. As expected, in this case, the method trains faster and has better performance, but still does not outperform BoN-*batch hard*. This experiment shows that BoN is a key component to the accelerated training and improved validation results of BoN-*batch hard*.

We implemented two methods for batch selection known in the literature, Hierarchical Tree (HT) [32] and 100k IDs [18], and combined them with *batch hard*. We followed the procedure described in Ge [32] and computed the distance matrix between all the IDs every 5k steps. We formed a batch by randomly selecting one ID, and taking the remaining $l-1$ as its closest neighbors. We trained a classifier on the whole train set for 10k steps and used this model to create the hash table with 10 bins. Additionally, we adapted one state-of-the-art hashing method [37] on image retrieval task for hard negative mining (see Section 6.3.2 for details). The results of all three methods confirm our hypothesis that batch sampling is important for improving and speeding up the training. However, none of them outperforms BoN neither in speed nor accuracy.

Even though BoN is specially designed to improve training of Siamese networks on large datasets, we tested the influence of BoN on two small datasets, CUB-200 [51] and Market-1501 [49]. BoN improves mAP on Market-1501 from 58.4 to 60.0, and from 36.1 to 37.9 on CUB-200. The improvement in these cases is smaller than in the experiments conducted on bigger datasets for two reasons: (1) *Batch hard* is usually enough, since the probability that hard samples exist in the mini-batch is higher than in case of large datasets; (2) Choosing optimal s becomes challenging: small s does not contain enough information for reconstruction, while bigger s leads to degenerate solution.

Table 4 shows the comparison of BoN-*batch hard* with state-of-the-art approaches on Stanford Online Products and DeepFashion In Shop datasets. We trained BoN-*batch hard* using the same training parameters as explained in Section 6.2, with a few changes: inception_v1 was used as backbone architecture (as in Oh Song et al. [7], Ge [32], Suh et al. [33], Kim et al. [52]) with an extra fully connected layer with frozen weights after the max pooling that reduces the embedding size to 256. We used images of size 336 $\times$ 336 pixels (as in Suh et al. [33]) with data augmentation techniques such as random horizontal flipping, blurring, zooming in and out and cutout. As the images in these datasets are more heterogeneous, the state-of-the-art methods usually do not use task specific architectures.

We show that BoN-*batch hard* provides better or comparable results than both [52], which uses attention ensembles, and stochastic class-based [33], which in addition to having higher complexity enhances its performance by using second order pooling [53], which introduces even more computational cost with respect to the baseline model. Additionally, BoN-*batch hard* performs better than DAML [30], which uses synthetic negative samples for training.

Our method achieves state-of-the-art results on Stanford Online Products dataset, while being comparable to the previously published methods evaluated on inShop dataset. The nature of Stanford Online Products dataset is more aligned with the problem that we are trying to solve: it has more training images than inShop (60k vs. 25.8k) as well as more classes (11.3k vs. 4k). We used the same $s = 10$ in both cases, so the hash table of the Stanford Online Products was more densely populated. Better performance would probably be obtained by training a model on inShop dataset with the smaller embedding size and smaller s.

8. Conclusion and future works

In this paper we introduced Bag of Negatives (BoN), a novel method for hard negative mining that accelerates and improves training of Siamese networks and scales well on datasets with large number of identities.

The main strengths of BoN are being computationally efficient and complementary to the popular *batch hard* approach. In fact, BoN provides a set of relevant negative samples, while *batch hard* provides the explicit hard negative selection process and the increased number of triplets per mini-batch; their combination provides improved validation results thanks to a better sampling of negative candidates. We also shown that BoN computational cost is negligible with respect to gradients computations during stochastic gradient descent based learning. It is also way more efficient than similar negative mining algorithms in the literature and it speeds-up the Spectral Hashing approach significantly. Summarising, BoN is better and faster than previous hard negative mining methods.

The main disadvantage of BoN is the requirement of a user provided s parameter. This parameter can be tuned by means of cross-validation or other standard meta-parameter tuning techniques. Nonetheless, we consider that an automatic strategy for tuning s would be very beneficial for the practical use of BoN on large datasets. For such a reason, future work will address possible solutions on automatic estimation of the s meta-parameter; since s must be a positive integer, one possible line of research is the simultaneous use of several values of s combined with an automated strategy of meta-parameter selection.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been partially supported by the Spanish project RTI2018-095645-B-C21, the grant 2018FI_B1_00056 from AGAUR and CERCA, from Generalitat de Catalunya.

References

[1] A. Gordo, J. Almazán, J. Revaud, D. Larlus, Deep image retrieval: learning global representations for image search, in: Proceedings of the European Conference on Computer Vision, Springer, 2016, pp. 241–257.
[2] F. Schroff, D. Kalenichenko, J. Philbin, FaceNet: a unified embedding for face recognition and clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823.
[3] L. Zhao, X. Li, Y. Zhuang, J. Wang, Deeply-learned part-aligned representations for person re-identification, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3219–3228.
[4] H. Zhao, M. Tian, S. Sun, J. Shao, J. Yan, S. Yi, X. Wang, X. Tang, Spindle Net: person re-identification with human body region guided feature decomposition and fusion, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1077–1085.
[5] C. Su, J. Li, S. Zhang, J. Xing, W. Gao, Q. Tian, Pose-driven deep convolutional model for person re-identification, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3960–3969.
[6] M. Saquib Sarfraz, A. Schumann, A. Eberle, R. Stiefelhagen, A pose-sensitive embedding for person re-identification with expanded cross neighborhood re-ranking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 420–429.
[7] H. Oh Song, Y. Xiang, S. Jegelka, S. Savarese, Deep metric learning via lifted structured feature embedding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4004–4012.
[8] W. Wu, D. Tao, H. Li, Z. Yang, J. Cheng, Deep features for person re-identification on metric learning, Pattern Recognition 110 (2020) 107424.
[9] M. Zhang, M. Xin, C. Gao, X. Wang, S. Zhang, Attention-aware scoring learning for person re-identification, Knowl. Based Syst. 203 (2020) 106154.
[10] A. Khatun, S. Denman, S. Sridharan, C. Fookes, Joint identification-verification for person re-identification: a four stream deep learning approach with improved quartet loss function, Computer Vision and Image Understanding 197-198 (2020) 102989.
[11] D. Manandhar, M. Bastan, K.-H. Yap, Semantic granularity metric learning for visual search, Journal of Visual Communication and Image Representation 72 (2020) 102871.
[12] X. Li, L. Yu, C.-W. Fu, M. Fang, P.-A. Heng, Revisiting metric learning for few-shot image classification, Neurocomputing 406 (2020) 49–58.
[13] H. Liu, J. Feng, M. Qi, J. Jiang, S. Yan, End-to-end comparative attention networks for person re-identification, IEEE Trans. Image Process. 26 (7) (2017) 3492–3506.
[14] W. Li, X. Zhu, S. Gong, Harmonious attention network for person re-identification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2285–2294.
[15] J. Xu, R. Zhao, F. Zhu, H. Wang, W. Ouyang, Attention-aware compositional network for person re-identification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2119–2128.
[16] Y. Guo, N.-M. Cheung, Efficient and deep person re-identification using multi-level similarity, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2335–2344.
[17] R.R. Varior, M. Haloi, G. Wang, Gated siamese convolutional neural network architecture for human re-identification, in: Proceedings of the IEEE European Conference on Computer Vision, Springer, 2016, pp. 791–808.
[18] C. Wang, X. Zhang, X. Lan, How to train triplet networks with 100k identities? in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2017, pp. 1907–1915.
[19] A. Hermans, L. Beyer, B. Leibe, In defense of the triplet loss for person re-identification, (2017) arXiv:1703.07737.
[20] Z. Zheng, L. Zheng, Y. Yang, Unlabeled samples generated by GAN improve the person re-identification baseline in vitro, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3754–3762.
[21] M.M. Kalayeh, E. Basaran, M. Gökmen, M.E. Kamasak, M. Shah, Human semantic parsing for person re-identification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1062–1071.
[22] I. Kemelmacher-Shlizerman, S.M. Seitz, D. Miller, E. Brossard, The megaface benchmark: 1 million faces for recognition at scale, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4873–4882.
[23] Z. Liu, P. Luo, S. Qiu, X. Wang, X. Tang, Deepfashion: powering robust clothes recognition and retrieval with rich annotations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1096–1104.
[24] C.-Y. Wu, R. Manmatha, A.J. Smola, P. Krahenbuhl, Sampling matters in deep embedding learning, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2840–2848.
[25] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, Y. Wu, Learning fine-grained image similarity with deep ranking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1386–1393.
[26] C. Wang, Q. Zhang, C. Huang, W. Liu, X. Wang, Mancs: a multi-task attentional network with curriculum sampling for person re-identification, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 365–381.
[27] Y. Xiao, J. Li, B. Du, J. Wu, J. Chang, W. Zhang, Memu: metric correlation siamese network and multi-class negative sampling for visual tracking, Pattern Recognit. 100 (2020) 107170.

[28] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, L. Song, Sphereface: deep hypersphere embedding for face recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 212–220.
[29] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, W. Liu, Cosface: large margin cosine loss for deep face recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5265–5274.
[30] Y. Duan, W. Zheng, X. Lin, J. Lu, J. Zhou, Deep adversarial metric learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2780–2789.
[31] S. Chen, C. Gong, J. Yang, X. Li, Y. Wei, J. Li, Adversarial metric learning, (2018) arXiv:1802.03170.
[32] W. Ge, Deep metric learning with hierarchical triplet loss, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 269–285.
[33] Y. Suh, B. Han, W. Kim, K.M. Lee, Stochastic class-based hard example mining for deep metric learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 7251–7259.
[34] B. Harwood, V. Kumar BG, G. Carneiro, I. Reid, T. Drummond, Smart mining for deep metric learning, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2821–2829.
[35] E. Yang, C. Deng, C. Li, W. Liu, J. Li, D. Tao, Shared predictive cross-modal deep quantization, IEEE Trans. Neural Netw. Learn. Syst. 29 (11) (2018) 5292–5303.
[36] L. Zhou, X. Bai, X. Liu, J. Zhou, E.R. Hancock, Learning binary code for fast nearest subspace search, Pattern Recognit. 98 (2020) 107040.
[37] C. Deng, E. Yang, T. Liu, J. Li, W. Liu, D. Tao, Unsupervised semantic-preserving adversarial hashing for image search, IEEE Trans. Image Process. 28 (8) (2019) 4032–4044.
[38] C. Deng, E. Yang, T. Liu, D. Tao, Two-stream deep hashing with class-specific centers for supervised image search, IEEE Trans. Neural Netw. Learn. Syst. 31 (6) (2019) 2189–2201.
[39] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: Advances in Neural Information Processing Systems, 2009, pp. 1753–1760.
[40] Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 35 (12) (2012) 2916–2929.
[41] M.A. Carreira-Perpinán, R. Raziperchikolaei, Hashing with binary autoencoders, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 557–566.
[42] H. Cardot, D. Degras, Online principal component analysis in high dimension: which algorithm to choose? Int. Stat. Rev. 86 (1) (2018) 29–50.
[43] W. Li, R. Zhao, X. Wang, Human reidentification with transferred metric learning, in: Proceedings of the Asian Conference on Computer Vision, Springer, 2012, pp. 31–44.
[44] W. Li, X. Wang, Locally aligned feature transforms across views, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 3594–3601.
[45] D. Baltieri, R. Vezzani, R. Cucchiara, 3DPes: 3D people dataset for surveillance and forensics, in: Proceedings of the 2011 Joint ACM Workshop on Human Gesture and Behavior Understanding, 2011, pp. 59–64.
[46] D. Gray, H. Tao, Viewpoint invariant pedestrian recognition with an ensemble of localized features, in: Proceedings of the European Conference on Computer Vision, Springer, 2008, pp. 262–275.
[47] S. Karanam, M. Gou, Z. Wu, A. Rates-Borras, O. Camps, R.J. Radke, A systematic evaluation and benchmark for person re-identification: features, metrics, and datasets, IEEE Trans. Pattern Anal. Mach. Intell. 41 (3) (2019) 523–536.
[48] L. Wei, S. Zhang, W. Gao, Q. Tian, Person transfer GAN to bridge domain gap for person re-identification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 79–88.
[49] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, Q. Tian, Scalable person re-identification: a benchmark, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1116–1124.
[50] E. Ristani, F. Solera, R. Zou, R. Cucchiara, C. Tomasi, Performance measures and a data set for multi-target, multi-camera tracking, in: Proceedings of the European Conference on Computer Vision, Springer, 2016, pp. 17–35.
[51] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, P. Perona, Caltech-UCSD Birds 200, Technical Report, California Institute of Technology, 2010.
[52] W. Kim, B. Goyal, K. Chawla, J. Lee, K. Kwon, Attention-based ensemble for deep metric learning, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 736–751.
[53] Y. Gao, O. Beijbom, N. Zhang, T. Darrell, Compact bilinear pooling, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 317–326.

Bojana Gajić received her B.Sc. and M.Sc. degrees in Electrical Engineering and Computer Science from University of Belgrade, Belgrade, Serbia, in 2014 and 2016. She is currently a Ph.D. candidate in the Department of Computer Science at the Autonomous University of Barcelona. During her Ph.D. she did internships in Xerox Research Center Europe, Naver Labs Europe and Vintra Inc. Her research interests are in the areas of image retrieval, person re-identification and face recognition.

Ariel Amato received the graduate degree in Electronics Engineer from Universidad Tecnológica Nacional, Córdoba, Argentina, in 2004, and the M.Sc. degree in computer vision and artificial intelligence from the Universitat Autónoma de Barcelona, Spain, in 2007; in 2012, received a Ph.D. degree in computer science with a focus on computer vision also from the Universitat Autònoma de Barcelona. From 2013 to 2015 he was a Marie Currie fellow at the Computer Vision Center (CVC) and the Department of Architecture Design and Media Technology of Aalborg University Copenhagen. Currently the Co-Founder and Chief Technology Officer at Vintra Inc., focused on bringing a new generation of augmented intelligence (the combination of machine and human intelligence) capabilities and tools to market.

Carlo Gatta obtained the degree in Electronic Engineering in 2001 from the Universitá degli Studi di Brescia (Italy). In 2006 he received the Ph.D. in Computer Science at the Universitàegli Studi di Milano (Italy), with a thesis on perceptually based color image processing. In September 2007 he joined the Computer Vision Center at Universitat Autonoma de Barcelona (UAB) as a postdoc researcher working mainly on medical imaging, computer vision and machine learning. He has been a member of the Computer Vision Center and the BCN Perceptual Computing Lab until April 2016. Currently, he works as a research scientist and machine learning team leader for Vintra Inc. His main research interests are deep learning, computer vision, machine learning, and image processing.