NetVLAD: CNN architecture for weakly supervised place recognition

Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic

Abstract—We tackle the problem of large scale visual place recognition, where the task is to quickly and accurately recognize the location of a given query photograph. We present the following four principal contributions. First, we develop a convolutional neural network (CNN) architecture that is trainable in an end-to-end manner directly for the place recognition task. The main component of this architecture, NetVLAD, is a new generalized VLAD layer, inspired by the "Vector of Locally Aggregated Descriptors" image representation commonly used in image retrieval. The layer is readily pluggable into any CNN architecture and amenable to training via backpropagation. Second, we create a new weakly supervised ranking loss, which enables end-to-end learning of the architecture's parameters from images depicting the same places over time downloaded from Google Street View Time Machine. Third, we develop an efficient training procedure which can be applied on very large-scale weakly labelled tasks. Finally, we show that the proposed architecture and training procedure significantly outperform non-learnt image representations and off-the-shelf CNN descriptors on challenging place recognition and image retrieval benchmarks.

1 Introduction

Visual place recognition has received a significant amount of attention in the past years both in computer vision [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11] and robotics communities [12], [13], [14], [15], [16] motivated by, e.g., applications in autonomous driving [14], augmented reality [17] or geo-localizing archival imagery [18].

The place recognition problem, however, still remains extremely challenging. How can we recognize the same street-corner in the entire city or on the scale of the entire country despite the fact it can be captured in different illuminations or change its appearance over time? The fundamental scientific question is what is the appropriate representation of a place that is rich enough to distinguish similarly looking places yet compact to represent entire cities or countries.

The place recognition problem has been traditionally cast as an instance retrieval task, where the query image location is estimated using the locations of the most visually similar images obtained by querying a large geotagged database [1], [2], [3], [8], [9], [10]. Each database image is represented using local invariant features [19] such as SIFT [20] that are aggregated into a fixed length vector representation for the entire image such as bag-of-visual-words [21], [22], VLAD [23], [24] or Fisher vector [25], [26]. The resulting representation is then usually compressed and efficiently indexed [21], [27]. The image database can be further augmented by 3D structure that enables recovery of accurate camera pose [4], [11], [28].

(a) Mobile phone query (b) Retrieved image of same place

Fig. 1. Our trained NetVLAD descriptor correctly recognizes the location (b) of the query photograph (a) despite the large amount of clutter (people, cars), changes in viewpoint and completely different illumination (night vs daytime). **Please see figure 7 and the appendix for more examples.**

In the last few years, convolutional neural networks (CNNs) [29], [30] have emerged as powerful image representations for various category-level recognition tasks such as object classification [31], [32], [33], [34], scene recognition [35] or object detection [36]. The basic principles of CNNs are known from 80's [29], [30] and the recent successes are a combination of advances in GPU-based computation power together with large labelled image datasets [31]. It has been shown that the trained representations are, to some extent, transferable between recognition tasks [32], [36], [37], [38], [39], and direct application of CNN representations trained for object classification [31] as black-box descriptor extractors has brought some improvements in performance on instance-level recognition tasks [40], [41], [42], [43], [44], [45], [46], [47]. In this work we investigate whether the performance can be further improved by CNN representations developed and trained directly for place recognition. This requires addressing the following four main challenges: First, what is a

- *R. Arandjelović, P. Gronat and J. Sivic are with Inria, WILLOW, Departement d'Informatique de l'École Normale Supérieure, PSL Research University, ENS/INRIA/CNRS UMR 8548, Paris, France.*
  E-mail: {Relja.Arandjelovic, Petr.Gronat, Josef.Sivic}@inria.fr
- *J. Sivic is also with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague.*
- *R. Arandjelović is currently at DeepMind.*
  E-mail: relja@google.com
- *A. Torii is with the Department of Mechanical and Control Engineering, Graduate School of Science and Engineering, Tokyo Institute of Technology, Tokyo, Japan.*
  E-mail: torii@ctrl.titech.ac.jp
- *T. Pajdla is with the Department of Cybernetics, Faculty of Electrical Engineering and with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague.*
  E-mail: pajdla@cvut.cz

good CNN architecture for place recognition? Second, how to gather sufficient amount of annotated data for the training? Third, how can we train the developed architecture in an end-to-end manner tailored for the place recognition task? Fourth, how to perform computationally efficient training in order to scale up to very large datasets? To address these challenges, we bring the following four innovations.

First, building on the lessons learnt from the current well performing hand-engineered object retrieval and place recognition pipelines [10], [23], [48], [49], we develop a convolutional neural network architecture for place recognition that aggregates mid-level (conv5) convolutional features extracted from the entire image into a compact fixed length vector representation amenable to efficient indexing. To achieve this, we design a new trainable generalized VLAD layer, NetVLAD, inspired by the Vector of Locally Aggregated Descriptors (VLAD) representation [24] that has shown excellent performance in image retrieval and place recognition. The layer is readily pluggable into any CNN architecture and amenable to training via backpropagation. The resulting aggregated representation is then compressed using Principal Component Analysis (PCA) to obtain the final compact descriptor of the image.

Second, to train the architecture for place recognition, we gather a large dataset of multiple panoramic images depicting the same place from different viewpoints over time from the Google Street View Time Machine. Such data is available for vast areas of the world, but provides only weak form of supervision: we know the two panoramas are captured at approximately similar positions based on their (noisy) GPS but we don't know which parts of the panoramas depict the same parts of the scene.

Third, we create a new loss function, which enables end-to-end learning of the architecture's parameters, tailored for the place recognition task from the weakly labelled Time Machine imagery. The loss function is also more widely applicable to other ranking tasks where large amounts of weakly labelled data are available.

Fourth, we develop an efficient learning procedure which can be applied on very large-scale weakly labelled tasks. It requires only a fraction of the computational time of a naive implementation thanks to improved data efficiency through hard negative mining, combined with an effective use of caching.

The resulting representation is robust to changes in viewpoint and lighting conditions, while simultaneously learns to focus on the relevant parts of the image such as the building façades and the skyline, while ignoring confusing elements such as cars and people that may occur at many different places. We show that the proposed architecture and training procedure significantly outperform non-learnt image representations and off-the-shelf CNN descriptors on challenging place recognition and image retrieval benchmarks.

1.1 Related work

While there have been many improvements in designing better image retrieval [22], [23], [24], [25], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62] and place recognition [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] systems, not many works have performed learning for these tasks. All relevant learning-based approaches fall into one or both of the following two categories: (i) learning for an auxiliary task (e.g. some form of distinctiveness of local features [2], [9], [12], [63], [64], [65], [66]), and (ii) learning on top of shallow hand-engineered descriptors that cannot be fine-tuned for the target task [2], [6], [7], [48], [67]. Both of these are in spirit opposite to the core idea behind deep learning that has provided a major boost in performance in various recognition tasks: end-to-end learning. We will indeed show in section 6.2 that training representations directly for the end-task, place recognition, is crucial for obtaining good performance.

Numerous works concentrate on learning better local descriptors or metrics to compare them [56], [59], [68], [69], [70], [71], [72], [73], [74], [75], but even though some of them show results on image retrieval, the descriptors are learnt on the task of matching local image patches, and not directly with image retrieval in mind. Some of them also make use of hand-engineered features to bootstrap the learning, i.e. to provide noisy training data [56], [59], [69], [70], [74].

Several works have investigated using CNN-based features for image retrieval. These include treating activations from certain layers directly as descriptors by concatenating them [43], [76], or by pooling [40], [41], [42], [45], [46], [47]. However, none of these works actually train the CNNs for the task at hand, but use CNNs as black-box descriptor extractors. One exception is the work of Babenko et al. [76] in which the network is fine-tuned on an auxiliary task of classifying 700 landmarks. However, again the network is not trained directly on the target retrieval task. Very recent works [77], [78], published after the first version of this paper [79], train CNNs end-to-end for image retrieval by making use of image correspondences obtained from structure-from-motion models, i.e. they rely on pre-existing image retrieval pipelines based on precise matching of RootSIFT descriptors, spatial verification and bundle adjustment.

Weyand et al. [80] proposed a CNN-based method for geolocalization by partitioning the Earth into cells and treating place recognition as a classification task. While providing impressive rough city/country level estimates of where a photo is taken, their method is not capable of providing several-meter accuracy place recognition that we consider here, as their errors are measured in tens and hundreds of kilometres. Finally, [81] and [82] performed end-to-end learning for different but related tasks of ground-to-aerial matching [82] and camera pose estimation [81].

2 Method overview

Building on the success of current place recognition systems (e.g. [1], [2], [3], [4], [5], [8], [9], [10], [11]), we cast place recognition as image retrieval. The query image with unknown location is used to visually search a large geotagged image database, and the locations of top ranked images are used as suggestions for the location of the query. This is generally done by designing a function f which acts as the "image representation extractor", such that given an image I_i it produces a fixed size vector $f(I_i)$. The function is used to extract the representations for the entire database $\{I_i\}$, which can be done offline, and to extract the query image representation $f(q)$, done online. The visual search is performed by finding the nearest database image to the query, either exactly or through fast approximate nearest neighbour search, by sorting images based on the Euclidean distance $d(q, I_i)$ between $f(q)$ and $f(I_i)$.

While previous works have mainly used hand-engineered image representations (e.g. $f(I)$ corresponds to extracting SIFT descriptors [20], followed by pooling into a bag-of-words vector [21] or a VLAD vector [24]), here we propose to learn

the representation $f(I)$ in an end-to-end manner, directly optimized for the task of place recognition. The representation is parametrized with a set of parameters θ and we emphasize this fact by referring to it as $f_\theta(I)$. It follows that the Euclidean distance $d_\theta(I_i, I_j) = \|f_\theta(I_i) - f_\theta(I_j)\|$ also depends on the same parameters. An alternative setup would be to learn the distance function itself, but here we choose to fix the distance function to be Euclidean distance, and to pose our problem as the search for the explicit feature map f_θ which works well under the Euclidean distance.

In section 3 we describe the proposed representation f_θ based on a new deep convolutional neural network architecture inspired by the compact aggregated image descriptors for instance retrieval. In section 4 we describe a method to learn the parameters θ of the network in an end-to-end manner using weakly supervised training data from the Google Street View Time Machine.

3 Deep architecture for place recognition

This section describes the proposed CNN architecture f_θ, guided by the best practices from the image retrieval community. Most image retrieval pipelines are based on (i) extracting local descriptors, which are then (ii) pooled in an orderless manner. The motivation behind this choice is that the procedure provides significant robustness to translation and partial occlusion. Robustness to lighting and viewpoint changes is provided by the descriptors themselves.

In order to learn the representation end-to-end, we design a CNN architecture that mimics this standard retrieval pipeline in an unified and principled manner with differentiable modules, as shown in figure 2. For step (i), we crop the CNN at the last convolutional layer and view it as a dense descriptor extractor. This has been observed to work well for instance retrieval [40], [41], [44] and texture recognition [83]. Namely, the output of the last convolutional layer is a $H \times W \times D$ map which can be considered as a set of D-dimensional descriptors extracted at $H \times W$ spatial locations. For step (ii) we design a new pooling layer inspired by the Vector of Locally Aggregated Descriptors (VLAD) [24] that pools extracted descriptors into a fixed image representation and its parameters are learnable via back-propagation. We call this new pooling layer "NetVLAD" layer and describe it in the next section.

3.1 NetVLAD: A Generalized VLAD layer (f_{VLAD})

Vector of Locally Aggregated Descriptors (VLAD) [24] is a popular descriptor pooling method for both instance level retrieval [24] and image classification [42]. It captures information about the statistics of local descriptors aggregated over the image. Whereas bag-of-visual-words [21], [84] aggregation keeps counts of visual words, VLAD stores the sum of residuals (difference vector between the descriptor and its corresponding cluster centre) for each visual word.

Formally, given N D-dimensional local image descriptors $\{\mathbf{x}_i\}$ as input, and K cluster centres ("visual words") $\{\mathbf{c}_k\}$ as VLAD parameters, the output VLAD image representation V is $D \times K$-dimensional. For convenience we will write V as a $D \times K$ matrix, but this matrix is reshaped into a vector and, after normalization, used as the image representation. The (j, k) element of V is computed as follows:

$$V(j,k) = \sum_{i=1}^{N} a_k(\mathbf{x}_i) \left(x_i(j) - c_k(j)\right), \tag{1}$$

where $x_i(j)$ and $c_k(j)$ are the j-th dimensions of the i-th descriptor and k-th cluster centre, respectively. $a_k(\mathbf{x}_i)$ denotes the membership of the descriptor $\mathbf{x}_i$ to k-th visual word, i.e. it is 1 if cluster $\mathbf{c}_k$ is the closest cluster to descriptor $\mathbf{x}_i$ and 0 otherwise. Intuitively, each D-dimensional column k of V records the sum of residuals $(\mathbf{x}_i - \mathbf{c}_k)$ of descriptors which are assigned to cluster $\mathbf{c}_k$. The matrix V is then L2-normalized column-wise (intra-normalization [23]), reshaped into a vector, and finally L2-normalized in its entirety [24].

In order to profit from years of wisdom produced in image retrieval, we propose to mimic VLAD in a CNN framework and design a trainable generalized VLAD layer, *NetVLAD*. The result is a powerful image representation trainable end-to-end on the target task (in our case place recognition). To construct a layer amenable to training via backpropagation, it is required that the layer's operation is differentiable with respect to all its parameters and the input. Hence, the key challenge is to make the VLAD pooling differentiable, which we describe next.

The source of discontinuities in VLAD is the hard assignment $a_k(\mathbf{x}_i)$ of descriptors $\mathbf{x}_i$ to clusters centres $\mathbf{c}_k$. To make this operation differentiable, we replace it with soft assignment of descriptors to multiple clusters

$$\bar{a}_k(\mathbf{x}_i) = \frac{e^{-\alpha\|\mathbf{x}_i - \mathbf{c}_k\|^2}}{\sum_{k'} e^{-\alpha\|\mathbf{x}_i - \mathbf{c}_{k'}\|^2}}, \tag{2}$$

which assigns the weight of descriptor $\mathbf{x}_i$ to cluster $\mathbf{c}_k$ according to their proximity, but relative to proximities to other cluster centres. $\bar{a}_k(\mathbf{x}_i)$ ranges between 0 and 1, with the highest weight assigned to the closest cluster centre. α is a parameter (positive constant) that controls the decay of the response with the magnitude of the distance. Note that for $\alpha \to +\infty$ this setup replicates the original VLAD exactly as $\bar{a}_k(\mathbf{x}_i)$ for the closest cluster would be 1 and 0 otherwise.

By expanding the squares in (2), it is easy to see that the term $e^{-\alpha\|\mathbf{x}_i\|^2}$ cancels between the numerator and the denominator resulting in a soft-assignment of the following form

$$\bar{a}_k(\mathbf{x}_i) = \frac{e^{\mathbf{w}_k^T \mathbf{x}_i + b_k}}{\sum_{k'} e^{\mathbf{w}_{k'}^T \mathbf{x}_i + b_{k'}}}, \tag{3}$$

where vector $\mathbf{w}_k = 2\alpha\mathbf{c}_k$ and scalar $b_k = -\alpha\|\mathbf{c}_k\|^2$. The final form of the NetVLAD layer is obtained by plugging the soft-assignment (3) into the VLAD descriptor (1) resulting in

$$V(j,k) = \sum_{i=1}^{N} \frac{e^{\mathbf{w}_k^T \mathbf{x}_i + b_k}}{\sum_{k'} e^{\mathbf{w}_{k'}^T \mathbf{x}_i + b_{k'}}} \left(x_i(j) - c_k(j)\right), \tag{4}$$

where $\{\mathbf{w}_k\}$, $\{b_k\}$ and $\{\mathbf{c}_k\}$ are sets of trainable parameters for each cluster k. Similarly to the original VLAD descriptor, the NetVLAD layer aggregates the first order statistics of residuals $(\mathbf{x}_i - \mathbf{c}_k)$ in different parts of the descriptor space weighted by the soft-assignment $\bar{a}_k(\mathbf{x}_i)$ of descriptor $\mathbf{x}_i$ to cluster k. Note however, that the NetVLAD layer has three independent sets of parameters $\{\mathbf{w}_k\}$, $\{b_k\}$ and $\{\mathbf{c}_k\}$, compared to just $\{\mathbf{c}_k\}$ of the original VLAD. This enables greater flexibility than the original VLAD, as explained in figure 3. Decoupling $\{\mathbf{w}_k, b_k\}$ from $\{\mathbf{c}_k\}$ has been proposed in [23] as a means to adapt the VLAD to a new dataset. All parameters of NetVLAD are learnt for the specific task in an end-to-end manner.

As illustrated in figure 2, the NetVLAD layer can be visualized as a meta-layer that is further decomposed into basic CNN layers

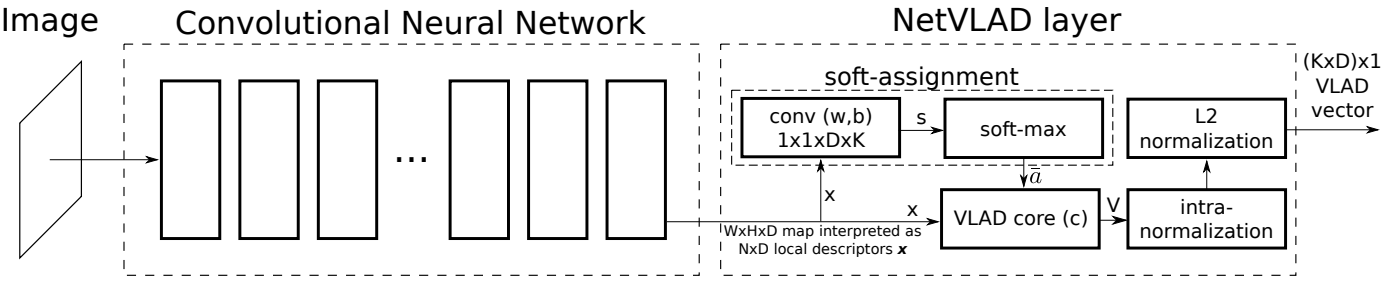


Fig. 2. **CNN architecture with the NetVLAD layer.** The layer can be implemented using standard CNN layers (convolutions, softmax, L2-normalization) and one easy-to-implement aggregation layer to perform aggregation in equation (4) ("VLAD core"), joined up in a directed acyclic graph. Parameters are shown in brackets.

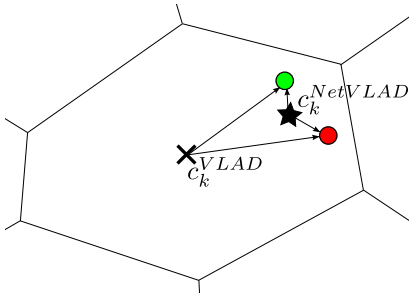


Fig. 3. **Benefits of supervised VLAD.** Red and green circles are local descriptors from two different images, assigned to the same cluster (Voronoi cell). Under the VLAD encoding, their contribution to the similarity score between the two images is the scalar product (as final VLAD vectors are L2-normalized) between the corresponding residuals, where a residual vector is computed as the difference between the descriptor and the cluster's anchor point. The anchor point $\mathbf{c}_k$ can be interpreted as the origin of a new coordinate system local to the the specific cluster k. In standard VLAD, the anchor is chosen as the cluster centre ($\times$) in order to evenly distribute the residuals across the database. However, in a supervised setting where the two descriptors are known to belong to images which should not match, it is possible to learn a better anchor ($\star$) which causes the scalar product between the new residuals to be small.

connected up in a directed acyclic graph. First, note that the first term in equation (4) is a soft-max function $\sigma_k(\mathbf{z}) = \frac{\exp(z_k)}{\sum_{k'} \exp(z_{k'})}$. Therefore, the soft-assignment of the input array of descriptors $\mathbf{x}_i$ into K clusters can be seen as a two step process: (i) a convolution with a set of K filters $\{\mathbf{w}_k\}$ that have spatial support 1×1 and biases $\{b_k\}$, producing the output $s_k(\mathbf{x}_i) = \mathbf{w}_k^T \mathbf{x}_i + b_k$; (ii) the convolution output is then passed through the soft-max function σ_k to obtain the final soft-assignment $\bar{a}_k(\mathbf{x}_i)$ that weights the different terms in the aggregation layer that implements equation (4). The output after normalization is a $(K \times D) \times 1$ descriptor.

Relations to other methods. Other works have proposed to pool CNN activations using VLAD or Fisher Vectors (FV) [42], [83], but do not learn the VLAD/FV parameters nor the input descriptors. The most related method to ours is the one of Sydorov et al. [85], which proposes to learn FV parameters jointly with an SVM for the end classification objective. However, in their work it is not possible to learn the input descriptors as they are hand-engineered (SIFT), while our VLAD layer is easily pluggable into any CNN architecture as it is amenable to backpropagation. "Fisher Networks" [86] stack Fisher Vector layers on top of each other, but the system is not trained end-to-end, only hand-crafted features are used, and the layers are trained greedily in a bottom-up fashion. Finally, our architecture is also related to bilinear networks [87], recently developed for a different task of fine-grained category-level recognition.

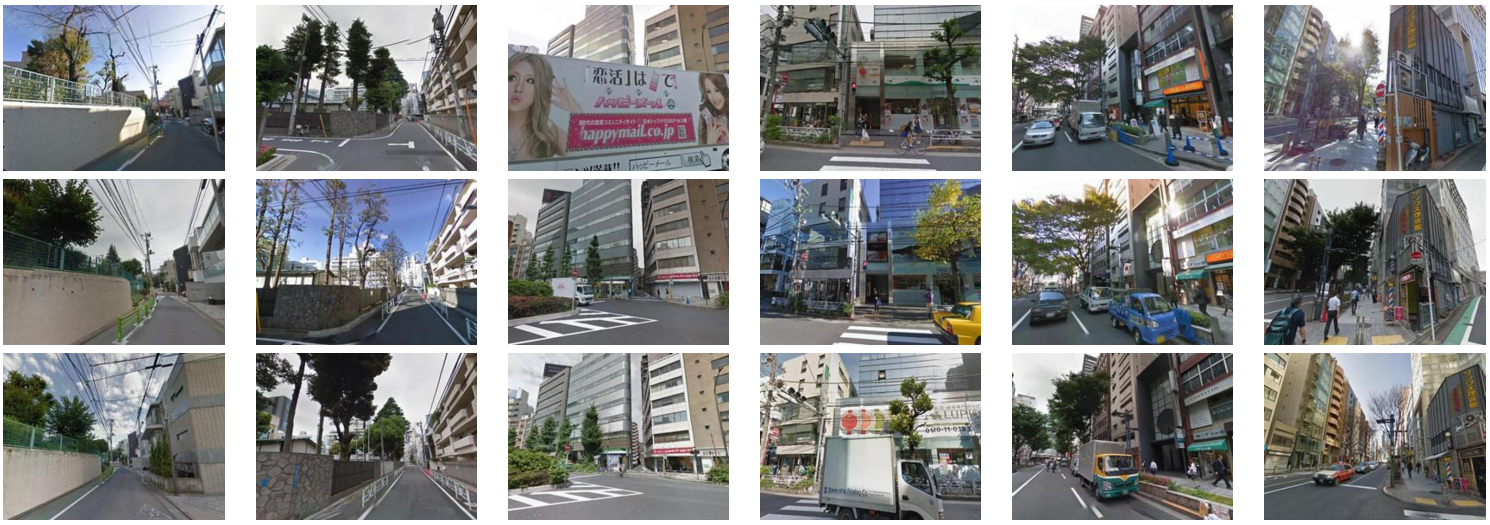
Max pooling (f_{max}). We also experiment with Max-pooling of the D-dimensional features across the $H \times W$ spatial locations, thus producing a D-dimensional output vector, which is then L2-normalized. Both of these operations can be implemented using standard layers in public CNN packages. This setup mirrors the method of [40], [44], but a crucial difference is that we will learn the representation (section 4) while [40], [43], [44] only use pretrained networks. Results will show (section 6.2) that simply using CNNs off-the-shelf [43] results in poor performance, and that training for the end-task is crucial. Additionally, VLAD will prove itself to be superior to the Max-pooling baseline.

4 Learning from Time Machine Data

In the previous section we have designed a new CNN architecture as an image representation for place recognition. Here we describe how to learn its parameters in an end-to-end manner for the place recognition task. The two main challenges are: (i) how to gather enough annotated training data and (ii) what is the appropriate loss for the place recognition task. To address theses issues, we will first show that it is possible to obtain large amounts of weakly labelled imagery depicting the same places over time from the Google Street View Time Machine. Second, we will design a new weakly supervised triplet ranking loss that can deal with the incomplete and noisy position annotations of the Street View Time Machine imagery. The details are below.

Weak supervision from the Time Machine. We propose to exploit a new source of data – Google Street View Time Machine – which provides multiple street-level panoramic images taken at different times at close-by spatial locations on the map. As will be seen in section 6.2, this novel data source is precious for learning an image representation for place recognition. As shown in figure 4, the same locations are depicted at different times and seasons, providing the learning algorithm with crucial information it can use to discover which features are useful or distracting, and what changes should the image representation be invariant to, in order to achieve good place recognition performance.

The downside of the Time Machine imagery is that it provides only incomplete and noisy supervision. Each Time Machine panorama comes with a GPS tag giving only its approximate location on the map (GPS accuracy of Google Street View has been estimated to 7–15 meters by [8]), which can be used to identify close-by panoramas but does not provide correspondences between parts of the depicted scenes. In detail, as the test queries



(a) (b) (c) (d) (e) (f)

Fig. 4. **Google Street View Time Machine examples.** Each column shows perspective images generated from panoramas from nearby locations, taken at different times. The goal of this work is to learn from this imagery an image representation that: has a degree of invariance to changes in viewpoint and illumination (a-f); has tolerance to partial occlusions (c-f); suppresses confusing visual information such as clouds (a,c), vehicles (c-f) and people (c-f); and chooses to either ignore vegetation or learn a season-invariant vegetation representation (a-f).

are perspective images from camera phones, each panorama is represented by a set of perspective images sampled evenly in different orientations and two elevation angles [2], [3], [6], [8]. Each perspective image is labelled with the GPS position of the source panorama. As a result, two geographically close perspective images do not necessarily depict the same objects since they could be facing different directions or occlusions could take place (e.g. the two images are around a corner from each other), etc. Thus, for a given training query q, the GPS information can only be used as a source of (i) *potential* positives $\{p_i^q\}$, i.e. images that are geographically close to the query, and (ii) *definite* negatives $\{n_j^q\}$, i.e. images that are geographically far from the query.[1]

Weakly supervised triplet ranking loss. We wish to learn a representation f_θ that will optimize place recognition performance. That is, for a given test query image q, the goal is to rank a database image I_{i*} from a close-by location higher than all other far away images I_i in the database. In other words, we wish the Euclidean distance $d_\theta(q, I)$ between the query q and a close-by image I_{i*} to be smaller than the distance to far away images in the database I_i, i.e. $d_\theta(q, I_{i*}) < d_\theta(q, I_i)$, for all images I_i further than a certain distance from the query on the map. Next we show how this requirement can be translated into a ranking loss between training triplets $\{q, I_{i*}, I_i\}$.

From the Google Street View Time Machine data, we obtain a training dataset of tuples $(q, \{p_i^q\}, \{n_j^q\})$, where for each training query image q we have a set of potential positives $\{p_i^q\}$ and the set of definite negatives $\{n_j^q\}$. The set of potential positives contains *at least one* positive image that should match the query, but we do not know which one. To address this ambiguity, we propose to identify the best matching potential positive image p_{i*}^q

$$p_{i*}^q = \underset{p_i^q}{\operatorname{argmin}}\, d_\theta(q, p_i^q) \tag{5}$$

1. Note that even faraway images can depict the same object. For example, the Eiffel Tower can be visible from two faraway locations in Paris. But, for the purpose of localization we consider in this paper such image pairs as negative examples because they are not taken from the same place.

for each training tuple $(q, \{p_i^q\}, \{n_j^q\})$. The goal then becomes to learn an image representation f_θ so that distance $d_\theta(q, p_{i*}^q)$ between the training query q and the best matching potential positive p_{i*}^q is smaller than the distance $d_\theta(q, n_j^q)$ between the query q and *all* negative images n_j^q:

$$d_\theta(q, p_{i*}^q) < d_\theta(q, n_j^q), \quad \forall j. \tag{6}$$

Based on this intuition we define a *weakly supervised ranking loss* L_θ for a training tuple $(q, \{p_i^q\}, \{n_j^q\})$ as

$$L_\theta = \sum_j l\left(\min_i d_\theta^2(q, p_i^q) + m - d_\theta^2(q, n_j^q)\right), \tag{7}$$

where l is the hinge loss $l(x) = \max(x, 0)$, and m is a constant parameter giving the margin. Note that equation (7) is a sum of individual losses for negative images n_j^q. For each negative, the loss l is zero if the distance between the query and the negative is greater by a margin than the distance between the query and the best matching positive. Conversely, if the margin between the distance to the negative image and to the best matching positive is violated, the loss is proportional to the amount of violation. Note that the above loss is related to the commonly used triplet loss [88], [89], [90], [91], but adapted to our weakly supervised scenario using a formulation (given by equation (5)) similar to multiple instance learning [92], [93], [94].

5 Efficient Large-Scale Training

In this section we discuss how to practically implement the weakly supervised learning method outlined in the previous section. Namely, we show that the naive implementation is prohibitively slow when training with large amounts of data, and present a training procedure which uses hard negative mining and caching to alleviate inefficiencies. We further improve upon this method by query clustering in order to share computations across training tuples. While down-scaling images can be used to make the training faster as the forward and backward passes require fewer

computations, two points should be noted: (i) our speedups are complementary to any improvements made to the speed of forward and backward passes, (ii) down-sampling images is not recommended as it causes a large decrease in recognition performance [45], [46].

5.1 Efficient hard negative mining with caching

A naive implementation of weakly supervised training from section 4 with Stochastic Gradient Descent (SGD) involves constructing training tuples $(q, \{p_i^q\}, \{n_j^q\})$ by randomly sampling training queries and adding their respective potential positives and definite negatives to the tuple. In order to compute the value of the loss and the derivative of the loss with respect to network parameters θ, needed to perform learning via back propagation, it is necessary to first obtain the image representations of all the images in the tuple, which involves executing the forward pass of the network for each image.

Inefficiencies of the naive approach. It is clearly intractable to include all the negatives in the tuple, as then the tuple would contain all the images in the training set. SGD would require recomputation of image representations for the entire set, which would be extremely inefficient. A commonly used straight forward solution is to simply randomly sample a set of negatives for the query each epoch[2]. However, a problem occurs in the case when there is a very large number of negatives and the image representation is "relatively good", but still far from its potential, so that most of the negatives are "easy". In this case, most of the negatives do not violate the margin and yield zero loss (equation (7)), causing zero gradients of the loss with respect to the parameters θ, in turn causing no updates to be made to θ. This leads to a vast waste of computational resources as most of the training time is spent on computing the image representations to evaluate the loss, only to find that the margin has not been violated and that the tuple does not provide useful information for training.

This is a real problem – for example, a simple baseline image representation, which performs Max pooling on top of a network pretrained on ImageNet, when searching in a database of 80k images, can retrieve a correct positive within the top 25 retrievals for more than 75% of the queries. While this performance is far from good (as will be seen in section 6.2), it means that for 75% of the queries, a randomly sampled negative is extremely unlikely (less than 25 out of 80k, i.e. 0.03%) to be ranked higher than the best positive, and therefore at least 75% of the computation is wasted because it causes no updates to the parameters.

Hard negative mining. We use *hard negative mining* [95], [96], [97] to solve this problem, namely, the negative set $\{n_j^q\}$ is compiled out of 10 hardest negatives for the query q, where the "hardness" of a negative is naturally measured as its contribution to the loss (7), i.e. how much it violates the margin. A natural question arises – how to choose these hard negatives? They can be precomputed once before training begins and the same hard negatives can be used throughout the training. We found that this simple strategy does not perform well as the network quickly learns how to solve the originally hard cases, without departing much from the initial network. Mixing the hard negatives with random negatives also falls short as, again, the original hard negatives are quickly solved and the method degenerates to the naive approach of random negative sampling. Therefore, we propose to keep track and update the current hardest negatives for each query. The negative set $\{n_j^q\}$ is compiled out of the 10 hardest negatives from a pool of (i) 1000 randomly sampled negatives and (ii) 10 hardest negatives from the previous epoch for query q.

The large number of randomly sampled negatives is required in order to bring sufficient variability into the negative pool. However, this yields the second inefficiency – at every step of SGD, to find the 10 hardest negatives, one still needs to compute 1010 image representations for the negative pool from which the hardest negatives are selected. Even for efficient GPU implementations, performing thousands of forward passes on full-resolution images at each SGD step creates a large computational bottleneck.

Caching. To address this issue, we introduce *caching* – image representations are computed for the entire training query and database sets, and are cached for a certain amount of time. The hard negative mining then uses these cached but slightly stale representations to obtain the 10 hardest examples and the forward and backward passes are only performed on these 10, compared to the original 1010, thus providing a huge computational saving. Furthermore, we also use the cached representations to select the best potential positive (equation (5)) as the number of potential positives for a query can also be relatively large (e.g. 50).

However, it is important to recompute the cached representations every once in a while. We have observed slow convergence if the cache is fixed for too long as the network quickly learns to be better than the fixed cache and then wastes time overfitting it. We found that recomputing the cached representations every 500 to 1000 training queries yields a good trade-off between epoch duration, convergence speed and quality of the solution. As described in section 6.1, we half the learning rate every 5 epochs – this causes the cached representations to change less rapidly, so we half the recomputation frequency every 5 epochs as well.

Computational complexity. Next, we compare the computational complexity of our methods. Let N, N_q, $N_{n,r}$, $N_{n,h}$ and N_p be the number of training database images, queries, random negatives, hard negatives, and average number of potential positives per query, respectively, and N_r the number of queries between the cache recomputation. Typical values of these quantities for our training datasets (table 1) and parameter choices are: $N = 80k$, $N_q = 8k$, $N_{n,r} = 1000$, $N_{n,h} = 10$, $N_p = 50$ and $N_r = 500$.

The main computational cost for the training is the execution of the forward and backward passes through the network, the former computing the image representation and the latter being used for network parameter updates. The speed of the forward and the backward pass is fairly comparable so we express the computational complexity of a learning method in terms of the combined number of times that the forward and the backward passes have to be executed in an epoch.

The number of backward passes, N_{backward}, is constant across all methods. For each query, it is necessary to perform a backward pass on the query, best potential positive, and violating hardest negatives (the number of which is upper bounded by $N_{n,h}$), i.e. the backward pass is computed at most N_{backward} times in an epoch, where N_{backward} equals:

$$N_{\text{backward}} = N_q \times (1 + 1 + N_{n,h}) \tag{8}$$

For typical values, $N_{\text{backward}} = 96k$.

Without caching (*hard+nocache*), for each query, the forward pass has to be executed for the query, all its potential positives,

2. We define one epoch as one pass through the training *queries*, not necessarily visiting all the training database images.

random negatives and current hard negatives, i.e. a total number of times is:

$$N_{\text{forward}}^{\text{hard+nocache}} = N_q \times (1 + N_p + N_{n,r} + N_{n,h}) \quad (9)$$

With caching (*hard+cache*), where the image representations are recomputed every N_r queries, i.e. N_q/N_r times in an epoch, the number of times the forward pass needs to be executed is:

$$N_{\text{forward}}^{\text{hard+cache}} = N_q/N_r \times N + N_q \times (1 + 1 + N_{n,h}) \quad (10)$$

Clearly, the relative speed of hard+cache versus hard+nocache depends on the characteristics of the training data and the parameter settings, but for our typical values listed above, hard+nocache requires $N_{\text{forward}}^{\text{hard+nocache}} + N_{\text{backward}} = 8.6$ million computations, while hard+cache this number drops to $N_{\text{forward}}^{\text{hard+cache}} + N_{\text{backward}} = 1.5$ million, giving a 5.7x speedup. While hard+cache takes 19 hours of training per epoch on a single GPU[3] (base network: VGG-16), hard+nocache is impractical for our purposes as it would require almost 5 days for a single epoch.

5.2 Scaling up further: query clustering

The techniques presented in the previous section are sufficient for our purposes when training sets include an order of 100k images; all the results (section 6.2) are obtained by using efficient hard negative mining with caching (hard+cache). However, scaling up to even large datasets is desired, as well as the ability to train networks faster even for the training set sizes used in this work.

The main bottleneck of hard+cache is the periodic recomputation of the cache for all images in the training set. With growing datasets, the recomputation becomes less feasible, e.g. revisiting the analysis of the computational complexity in the previous section, if the database grows 10 times (i.e. $N = 800k$), the caching would actually be slower than not caching. In the following, we present a method, hard+cache+query, which uses caching but its computational complexity does not depend on the database size, providing speedups over both hard+cache and hard+nocache.

First, we investigate what information is required by the weakly supervised training for a block of N_r training queries: (i) the N_r query images, (ii) N_p potential positives for each of the N_r queries. (iii) $N_{n,h}$ current hardest negatives for each of the N_r queries, and (iv) $N_{n,r}$ random negatives for each of the N_r queries. While (i) and (iii) are unavoidable, (ii) and (iv) can be improved upon by sharing as much information as possible across the N_r queries.

The random negative pool can be shared across the N_r queries, i.e. instead of having a different set of $N_{n,r}$ negatives for each query, we keep the random pool of size $N_{n,r}$ fixed for N_r consecutive queries. This strategy brings down the computational complexity to $N_q/N_r \times N_{n,r} + N_q \times (1 + N_p + N_{n,h}) + N_{\text{backward}}$. The potential positives term $N_q \times N_p$ becomes the new bottleneck, limiting the speedup versus hard+cache to only 2.5x. At first sight, it does not seem possible to decrease this term as it is essential that each training tuple contains all potential positives, in order to ensure the validity of weakly supervised training.

The key idea that enables a further speedup is to group the queries such that their potential positives overlap, therefore decreasing the forward pass count to:

$$N_{\text{forward}}^{\text{hard+cache+query}} = N_q/N_r \times N_{n,r} + N_q \times (1 + N_{p,e} + N_{n,h}), \quad (11)$$

where $N_{p,e}$ is the *effective* number of potential positives whose image representations have to be computed per query, i.e. the total number of potential positives for all queries in the group divided by the size of the query group. The grouping is accomplished by clustering the training queries according to their GPS location, and processing queries in the order of their cluster membership. Note that to emulate SGD where queries are traversed in a randomized order, each epoch the queries are reclustered starting with a different random seed, and the cluster order as well as within cluster query order are randomized as well.

Computational complexity. The total number of forward and backward passes that have to be computed in an epoch for our hard+cache+query method is $N_{\text{forward}}^{\text{hard+cache+query}} + N_{\text{backward}}$. In our running example typical training scenario, $N_{p,e}$ is measured to be 5, and the above quantity amounts to a mere 240k, giving a theoretical speedup versus hard+cache of 6.2x, and 36x over hard+nocache. In practice, hard+cache+query only takes 3.5 hours for an epoch, compared to 19 hours of hard+cache, providing a real 5.4x speedup. Note that the query location-based clustering takes next to no time as it only requires clustering of a small number of 2-D data points (in our case $N_q = 8k$), but the method would have no problems scaling to orders of magnitude more queries, and if needed one could also resort to approximate k-means clustering [22].

Discussion. The presented hard+cache+query method enables efficient large-scale training, and is substantially faster than both hard+cache and hard+nocache, irrespective of the dataset statistics such as the size of the the training database or the number of training queries. In terms of end place recognition performance, we have not observed any statistically significant differences compared to hard+cache. Hard+cache+query is the only method capable of scaling up to ever increasing training datasets, and we believe that training with even more data will produce better models, as this is a consistently observed trend in deep learning-based methods. One potential downside of the method, although we have not observed its effects, is that, unlike hard+cache/nocache, consecutive training tuples are correlated due to the shared random negatives and overlapping potential positives, which is known to be detrimental when training with SGD. It is likely that applying simple and standard schemes such as experience replay [98], [99] or parallel training with asynchronous updates [100] would be beneficial as they are able to effectively counter correlated sequences of training data.

6 EXPERIMENTS

In this section we describe the used datasets and evaluation methodology (section 6.1), and give quantitative and qualitative (section 6.2) results to validate our approach. Finally, we also test the method on the standard image retrieval benchmarks (section 6.3).

3. This is the time for the first epoch – the later epochs tend to be faster as (i) fewer negatives violate the margin so fewer backward passes and parameter updates are required (speedup of second versus first epoch is 1.3x), and (ii) the frequency of cache recomputation decreases, as described in the previous section.

TABLE 1
Datasets. Sizes of datasets used in experiments. All train/val(idation)/test datasets are mutually disjoint geographically.

Dataset	Database	Query set
Pitts250k-train	91,464	7,824
Pitts250k-val	78,648	7,608
Pitts250k-test	83,952	8,280
Pitts30k-train	10,000	7,416
Pitts30k-val	10,000	7,608
Pitts30k-test	10,000	6,816
Tokyo Time Machine-train	49,104	7,277
Tokyo Time Machine-val	49,056	7,186
Tokyo 24/7 (=test)	75,984	315

6.1 Datasets and evaluation methodology

We report results on two publicly available place recognition datasets.

Pittsburgh (Pitts250k) [8] contains 250k database images downloaded from Google Street View and 24k test queries generated from Street View but taken at different times, years apart. We divide this dataset into three roughly equal parts for training, validation and testing, each containing around 83k database images and 8k queries, where the division was done geographically to ensure the sets contain independent images. To facilitate faster training, for some experiments, a smaller subset (Pitts30k) is used, containing 10k database images in each of the train/val(idation)/test sets, which are also geographically disjoint.

Tokyo 24/7 [10] contains 76k database images and 315 query images taken using mobile phone cameras. This is an extremely challenging dataset where the queries were taken at daytime, sunset and night, while the database images were only taken at daytime as they originate from Google Street View as described above. To form the train/val sets we collected additional Google Street View panoramas of Tokyo using the Time Machine feature, and name this set **TokyoTM**; Tokyo 24/7 (=test) and TokyoTM train/val are all geographically disjoint. The newly collected TokyoTM database (provided on request) was generated from Time Machine panoramas, downloaded using [101], such that each panorama is represented by a set of 12 perspective images sampled evenly in different orientations [2], [3], [6], [8], [10]. Figure 4 shows samples from the dataset. The query set is formed as a random subsample of the database, and for each query, the positive and negative sets are sampled from the database so that they have a time stamp at least one month away from the time stamp of the query. This is done for both training and validation sets.

Table 1 shows the sizes of the datasets. All images are 640×480 apart from the queries of Tokyo 24/7, which are, following [10], resized such that their largest dimension is 640.

Evaluation metric. We follow the standard place recognition evaluation procedure [5], [6], [8], [9], [10]. The query image is deemed correctly localized if at least one of the top N retrieved database images is within $d = 25$ meters from the ground truth position of the query. The percentage of correctly recognized queries (Recall) is then plotted for different values of N. For Tokyo 24/7 we follow [10] and perform spatial non-maximal suppression on ranked database images before evaluation.

Implementation details. We use two base architectures which are extended with Max pooling (f_{max}) and our NetVLAD (f_{VLAD}) layers: AlexNet [31] and VGG-16 [33]; both are cropped at the last convolutional layer (conv5), before ReLU. For Max pooling we use raw conv5 descriptors (with no normalization) while for VLAD and NetVLAD we add an additional descriptor-wise L2-normalization layer after conv5. We found that not normalizing for Max pooling and normalizing for VLAD/NetVLAD generalizes across architectures, i.e. these are the best configurations for both AlexNet and VGG-16; more details are given in the appendix.

The number of clusters used in all VLAD / NetVLAD experiments is $K = 64$, resulting in 16k and 32k dimensional image representations for the two base architectures, AlexNet and VGG-16, respectively. The NetVLAD layer parameters are initialized to reproduce the conventional VLAD vectors by clustering conv5 descriptors extracted from a subsample of the train set for each dataset. The α parameter used for initialization is chosen to be large, such that the soft assignment weights $\bar{a}_k(\mathbf{x}_i)$ are very sparse in order to mimic the conventional VLAD well. Specifically, α is computed so that the the ratio of the largest and the second largest soft assignment weight $\bar{a}_k(\mathbf{x}_i)$ is on average equal to 100.

We use the margin $m = 0.1$, learning rate 0.001 or 0.0001 depending on the experiment, which is halved every 5 epochs, momentum 0.9, weight decay 0.001, batch size of 4 tuples (a tuple contains many images, c.f. equation (7)), and train for at most 30 epochs but convergence usually occurs much faster. The network which yields the best recall@5 on the validation set is used for testing.

All networks were trained using hard negative mining and caching (hard+cache), with the same parameter settings as in section 5. Query clustering (hard+cache+query) was not used in these experiments as we did not observe significant differences in the end performance of hard+cache+query versus hard+cache, and in order not to waste resources since we already performed the full set of experiments with hard+cache before establishing hard+cache+query.

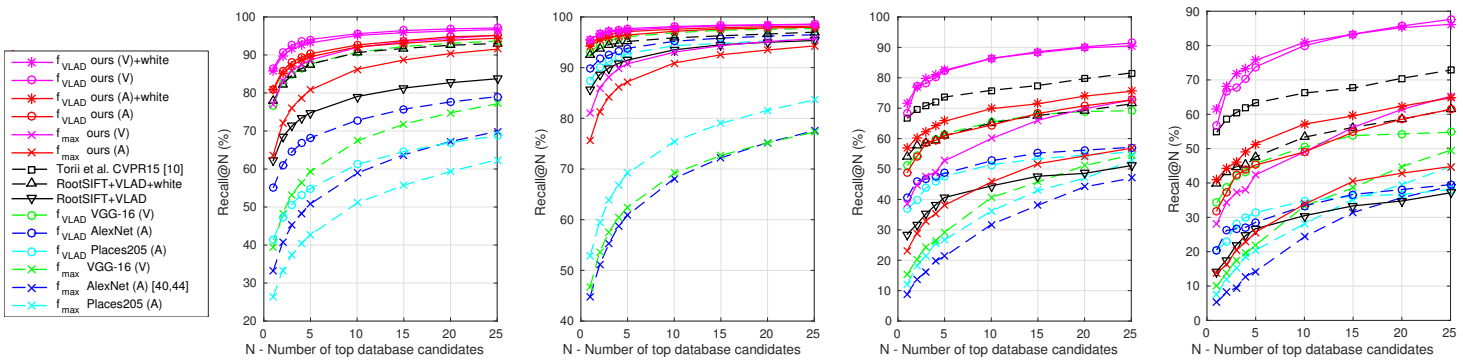
As the VGG-16 network is much deeper and more GPU-memory hungry than AlexNet, it was not possible to train it in its entirety. Instead, in the light of experiments in table 2, the VGG-16 network is only trained down to conv5 layer.

All training and evaluation code, as well as our trained networks, are online [102], implemented in the MatConvNet framework [103]. Additional tuning of parameters and jittering could further improve performance as we have still observed some amount of overfitting.

6.2 Results and discussion

Baselines and state-of-the-art. To assess benefits of our approach we compare our representations trained for place recognition against "off-the-shelf" networks pretrained on other tasks. Namely, given a base network cropped at conv5, the baselines either use Max pooling (f_{max}), or aggregate the descriptors into VLAD (f_{VLAD}), but perform no further task-specific training. The three base networks are: AlexNet [31], VGG-16 [33], both are pretrained for ImageNet classification [104], and Places205 [35], reusing the same architecture as AlexNet but pretrained for scene classification [35]. Pretrained networks have been recently used as off-the-shelf dense descriptor extractors for instance retrieval [40], [41], [42], [43], [44] and the untrained f_{max} network corresponds to the method of [40], [44].

Furthermore we compare our CNN representations trained for place recognition against the state-of-the-art local feature based compact descriptor, which consists of VLAD pooling [24] with



(a) Pitts250k-test (b) TokyoTM-val (c) Tokyo 24/7 all queries (d) Tokyo 24/7 sunset/night

Fig. 5. **Comparison of our methods versus off-the-shelf networks and state-of-the-art.** The base CNN architecture is denoted in brackets: (A)lexNet and (V)GG-16. Trained representations (red and magenta for AlexNet and VGG-16) outperform by a large margin off-the-shelf ones (blue, cyan, green for AlexNet, Places205, VGG-16), f_{VLAD} (-o-) works better than f_{max} (-x-), and our f_{VLAD}+whitening (-$*$-) representation based on VGG-16 sets the state-of-the-art on all datasets. [10] only evaluated on Tokyo 24/7 as the method relies on depth data not available in other datasets.

intra-normalization [23] on top of densely extracted RootSIFTs [20], [48]. The descriptor is optionally reduced to 4096 dimensions using PCA (learnt on the training set) combined with whitening and L2-normalization [49]; this setup together with view synthesis yields the state-of-the-art results on the challenging Tokyo 24/7 dataset (c.f. [10]).

In the following we discuss figure 5, which compares place recognition performance of our method to the baselines outlined above on the Pittsburgh and Tokyo 24/7 benchmarks.

Benefits of end-to-end training for place recognition. Representations trained on the end-task of place recognition consistently outperform by a large margin off-the-shelf CNNs on both benchmarks. For example, on the Pitts250k-test our trained AlexNet with (trained) NetVLAD aggregation layer achieves recall@1 of 81.0% compared to only 55.0% obtained by off-the-shelf AlexNet with standard VLAD aggregation, i.e. a relative improvement in recall of 47%. Similar improvements can be observed on all three datasets. This confirms two important premises of this work: (i) our approach can learn rich yet compact image representations for place recognition, and (ii) the popular idea of using pretrained networks "off-the-shelf" [40], [41], [42], [43], [44] is sub-optimal as the networks trained for object or scene classification are not necessary suitable for the end-task of place recognition. The failure of the "off-the-shelf networks" is not surprising – apart from the obvious benefits of training, it is not clear why it should be meaningful to directly compare conv5 activations using Euclidean distance as they are trained to be part of the network architecture. For example, one can insert an arbitrary affine transformation of the features that can be countered by the following fully connected layer (fc6). This is not a problem when transferring the pre-trained representation for object classification [32], [39] or detection [36] tasks, as such transformation can be countered by the follow-up adaptation [32] or classification [36], [39] layers that are trained for the target task. However, this is not the case for retrieval [40], [41], [42], [43], [44] when Euclidean distance is applied directly on the output "off-the-shelf" descriptors.

Dimensionality reduction. We follow the standard state-of-the-art procedure to perform dimensionality reduction of VLAD, as described earlier, i.e. the reduction into 4096-D is performed using PCA with whitening followed by L2-normalization [10], [49]. Figure 5 shows that the lower dimensional f_{VLAD} (-$*$-) performs similarly to the full size vector (-o-).

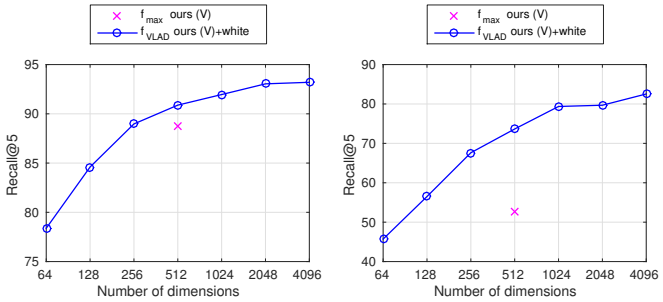
Another option is to add a fully connected (FC) layer on top of the network, followed by L2-normalization, and train the entire network end-to-end along with dimensionality reduction. We have found this approach to work as well as whitening, but to be sensitive to initialization (random orthogonal projection worked best) due to a large overfitting potential as the FC contains a huge number of parameters.

Comparison with state-of-the-art. Figure 5 also shows that our trained f_{VLAD} representation with whitening based on VGG-16 (magenta -$*$-) convincingly outperforms RootSIFT+VLAD+whitening, as well as the method of Torii et al. [10], and therefore sets the state-of-the-art for compact descriptors on all benchmarks. Note that these are strong baselines that outperform most off-the-shelf CNN descriptors on the place recognition task.

NetVLAD versus Max pooling. By comparing f_{VLAD} (-o-) methods with their corresponding f_{max} (-x-) counterparts it is clear that NetVLAD pooling is much better than Max pooling for both off-the-shelf and trained representations. Figure 6 shows that NetVLAD performance decreases gracefully with dimensionality: On Tokyo 24/7, 128-D NetVLAD performs similarly to 512-D Max pooling, resulting in *four* times more compact representation for the same performance. Similarly, on Pitts250k-test NetVLAD achieves a two-fold memory saving compared to Max pooling. Furthermore, NetVLAD+whitening outperforms Max pooling convincingly when reduced to the same dimensionality.

Max versus Sum. Recent work [41] suggests that Sum pooling performs better than Max pooling. Indeed, in our experiments Sum outperforms Max in the off-the-shelf set-up (recall@5 on Pitts250k-test – Sum: 67.9%, Max: 59.3%), but only for VGG-16, not AlexNet. Our training also works for Sum getting a significant improvement over the off-the-shelf set-up (+21% relative), but after training Max still performs better than Sum (Max: 88.7%, Sum: 82.3%). We also experimented with adding whitening to the Max-pooling baseline, but observed a drop in performance; detailed discussion is available in the appendix.

Is pooling needed? Another alternative image representation is to use the raw conv activations without any pooling, potentially



(a) Pitts250k-test (b) Tokyo 24/7

Fig. 6. **Place recognition accuracy versus dimensionality.** Note the log scale of the x-axis. 128-D NetVLAD performs comparably to the $4\times$ larger 512-D f_{max} on Tokyo 24/7. For the same dimension (512-D) NetVLAD convincingly outperforms f_{max}.

TABLE 2
Partial training. Effects of performing backpropagation only down to a certain layer of AlexNet, e.g. 'conv4' means that weights of layers from conv4 and above are learnt, while weights of layers below conv4 are fixed to their pretrained state; r@N signifies recall@N. Results are shown on the Pitts30k-val dataset.

Lowest trained layer	f_{max}			f_{VLAD}		
	r@1	r@5	r@10	r@1	r@5	r@10
none (off-the-shelf)	33.5	57.3	68.4	54.5	69.8	76.1
NetVLAD	—	—	—	80.5	91.8	95.2
conv5	63.8	83.8	89.0	84.1	94.6	95.5
conv4	62.1	83.6	89.2	85.1	94.4	96.1
conv3	**69.8**	86.7	90.3	**85.5**	94.6	96.5
conv2	69.1	**87.6**	**91.5**	84.5	94.6	**96.6**
conv1 (full)	68.5	86.2	90.8	84.2	**94.7**	96.1

followed with a fully connected (FC) layer. There are multiple issues with this approach: (i) it cannot handle variable image sizes, (ii) it would be infeasibly large, and, most importantly, (iii) it is not invariant to translation, scale or partial occlusions as it enforces a fixed spatial layout. Our experiments on Tokyo 24/7 confirm that pooling is indeed required, as the best trained "no-pool" baseline achieves a recall@5 of 63.5%, compared to 82.5% of NetVLAD. Further details are given in the appendix.

Which layers should be trained? In Table 2 we study the benefits of training different layers for the end-task of place recognition. The largest improvements are thanks to training the NetVLAD layer, but training other layers results in further improvements, with some overfitting occurring below conv2.

Pretraining. We investigated whether pretraining the convolutional layers with Max-pooling can be beneficial for the subsequent NetVLAD training. While the pretraining helped NetVLAD to train faster, the final performance was unaffected; details of the experiment are available in the appendix.

Importance of Time Machine training. Here we examine whether the network can be trained without the Time Machine (TM) data. In detail, we have modified the training query set for Pitts30k-train to be sampled from the same set as the training database images, i.e. the tuples of query and database images used in training were captured at the same time. As shown in table 3, Recall@1 with f_{max} on Pitts30k-val for the off-the-shelf AlexNet is 33.5%, and training without TM improves this to 38.7%. However, training with TM obtains 68.5% showing that

TABLE 3
Time Machine importance. Recall of f_{max} on Pitts30k-val (AlexNet) with vs without using Time Machine data for training. Training using Time Machine is essential for generalization.

Training data	recall@1	recall@10
Pretrained on ImageNet [31]	33.5	68.5
Pretrained on Places205 [35]	24.8	54.4
Trained without Time Machine	38.7	68.1
Trained with Time Machine	**68.5**	**90.8**

Time Machine data is crucial for good place recognition accuracy as without it the network does not generalize well. The network learns, for example, that recognizing cars is important for place recognition, as the same parked cars appear in all images of a place.

Qualitative evaluation. Figure 7 compares the top ranked images of our method versus the best baseline (RootSIFT+VLAD+whitening); additional examples are shown in the appendix. The performance of our descriptor is impressive – it can recognize the same place despite large changes in appearance due to illumination (day/night), viewpoint and partial occlusion by cars, trees and people. The main failure cases are queries taken at night which are very dark and featureless, causing problems even for humans. Top ranked images for all queries in the Tokyo 24/7 dataset are available on our project page [102].

6.3 Image retrieval

We use our best performing network (VGG-16, f_{VLAD} with whitening and dimensionality reduction down to 256-D) trained completely on Pittsburgh, to extract image representations for standard object and image retrieval benchmarks (Oxford 5k & 105k [22], Paris 6k [58], Holidays [53]). Table 4 compares NetVLAD to the state-of-the-art compact image representations (256-D). Our representation achieves the best mAP on Oxford and Paris by a large margin, e.g. +20% relative improvement on Oxford 5k (crop). It also sets the state-of-the-art on Holidays, but here training is detrimental as the dataset is less building-oriented (e.g. it also contains paysages, underwater photos, boats, cars, bears, etc), while our training only sees images from urban areas. We believe training on data more diverse than Pittsburgh streets can further improve performance. The complete set of NetVLAD results for different output dimensions is shown in the appendix.

Discussion of latest work. Recently, after publication of the first version of this work [79], [77] and [78] achieved even better results on the image retrieval tasks by using stronger supervision in the form of automatically cleaned-up image correspondences obtained with structure-from-motion, i.e. precise matching of RootSIFT descriptors and spatial verification. Note that the representation used in [78] is identical to our Max pooling baseline, which we show is inferior to NetVLAD in our setting. It would be interesting to see what performance would NetVLAD reach with strong supervision, but this comparison is beyond the scope of this paper.

Competitive performance has been also obtained by recent off-the-shelf methods: Tolias et al. [45] pool at different scales, while Kalantidis et al. [46] find that picking pool5 features compared to conv5 brings a boost, and perform image-adaptive descriptor weighting. These improvements are complementary to

Fig. 7. **Examples of retrieval results for challenging queries on Tokyo 24/7.** Each column corresponds to one test case: the query is shown in the first row, the top retrieved image using our best method (trained VGG-16 NetVLAD + whitening) in the second, and the top retrieved image using the best baseline (RootSIFT + VLAD + whitening) in the last row. The green and red borders correspond to correct and incorrect retrievals, respectively. Note that our learnt descriptor can recognize the same place despite large changes in appearance due to illumination (day/night), viewpoint and partial occlusion by cars, trees and people.

our NetVLAD architecture and can be incorporated in our end-to-end learning framework. Additional performance boosts can be achieved by incorporating a centrality prior [41], [47]. Mohedano et al. [47] pool descriptors using bag-of-words and thus their results are not directly comparable to our compact 256-D image representation, but it would be interesting to investigate if it is possible to train bag-of-words pooling in an end-to-end manner similarly to how we adapted VLAD to NetVLAD. Finally, Kim et al. [105] add another component to NetVLAD, which produces per-spatial location weights and performs weighted NetVLAD aggregation to achieve superior results.

7 Conclusions

We have designed a new convolutional neural network architecture that is trained for place recognition in an end-to-end manner from weakly supervised Street View Time Machine data. Our trained representation significantly outperforms non-learnt image representations and off-the-shelf CNN descriptors on challenging place recognition and image retrieval benchmarks. The two main components of our architecture – (i) the NetVLAD pooling layer and (ii) weakly supervised ranking loss – are generic CNN building blocks applicable beyond the place recognition task. The NetVLAD layer offers a powerful pooling mechanism with learnable parameters that can be easily plugged into any other CNN architecture. The weakly supervised ranking loss opens up the possibility of end-to-end learning for other ranking tasks where large amounts of weakly labelled data are available, for example, images described with natural language [107].

Acknowledgments

This work was partly supported by RVO13000 - Conceptual development of research organization, ERC grant LEAP (no. 336845), ANR project Semapolis (ANR-13-CORD-0003), JSPS KAKENHI Grant Number 15H05313, the Inria CityLab IPL, CIFAR Learning in Machines & Brains program, the EU-H2020 project LADIO No. 731970, the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory, contract FA8650-12-C-7212, and ESIF, OP Research, development and education project IMPACT No. $CZ.02.1.01/0.0/0.0/15\_003/0000468$. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

References

[1] G. Schindler, M. Brown, and R. Szeliski, "City-scale location recognition," in *Proc. CVPR*, 2007.
[2] J. Knopp, J. Sivic, and T. Pajdla, "Avoiding confusing features in place recognition," in *Proc. ECCV*, 2010.
[3] D. M. Chen, G. Baatz, K. Koeser, S. S. Tsai, R. Vedantham, T. Pylvanainen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk, "City-scale landmark identification on mobile devices," in *Proc. CVPR*, 2011.
[4] T. Sattler, B. Leibe, and L. Kobbelt, "Fast image-based localization using direct 2D–to–3D matching," in *Proc. ICCV*, 2011.
[5] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt, "Image retrieval for image-based localization revisited," in *Proc. BMVC.*, 2012.
[6] P. Gronat, J. Sivic, G. Obozinski, and T. Pajdla, "Learning and calibrating per-location classifiers for visual place recognition," *IJCV*, vol. 118, no. 3, pp. 319–336, 2016.
[7] S. Cao and N. Snavely, "Graph-based discriminative learning for location recognition," in *Proc. CVPR*, 2013.
[8] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi, "Visual place recognition with repetitive structures," in *Proc. CVPR*, 2013.
[9] R. Arandjelović and A. Zisserman, "DisLocation: Scalable descriptor distinctiveness for location recognition," in *Proc. ACCV*, 2014.
[10] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla, "24/7 place recognition by view synthesis," in *Proc. CVPR*, 2015.
[11] T. Sattler, M. Havlena, F. Radenović, K. Schindler, and M. Pollefeys, "Hyperpoints and fine vocabularies for large-scale location recognition," in *Proc. ICCV*, 2015.
[12] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, 2008.
[13] ——, "Highly scalable appearance-only SLAM - FAB-MAP 2.0," in *RSS*, 2009.
[14] C. McManus, W. Churchill, W. Maddern, A. Stewart, and P. Newman, "Shady dealings: Robust, long-term visual localisation using illumination invariance," in *Proc. Intl. Conf. on Robotics and Automation*, 2014.
[15] W. Maddern and S. Vidas, "Towards robust night and day place recognition using visible and thermal imaging," in *Proc. Intl. Conf. on Robotics and Automation*, 2014.
[16] N. Sunderhauf, S. Shirazi, A. Jacobson, E. Pepperell, F. Dayoub, B. Upcroft, and M. Milford, "Place recognition with ConvNet landmarks: Viewpoint-robust, condition-robust, training-free," in *RSS*, 2015.
[17] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt, "Scalable 6-DOF localization on mobile devices," in *Proc. ECCV*, 2014.
[18] M. Aubry, B. C. Russell, and J. Sivic, "Painting-to-3D model alignment via discriminative visual elements," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 2, p. 14, 2014.
[19] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Foundations and Trends® in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2008.
[20] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.
[21] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. ICCV*, vol. 2, 2003, pp. 1470–1477.
[22] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. CVPR*, 2007.
[23] R. Arandjelović and A. Zisserman, "All about VLAD," in *Proc. CVPR*, 2013.
[24] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proc. CVPR*, 2010.
[25] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Proc. CVPR*, 2010.
[26] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local images descriptors into compact codes," *IEEE PAMI*, 2012.
[27] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE PAMI*, 2011.
[28] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua, "Worldwide pose estimation using 3D point clouds," in *Proc. ECCV*, 2012.
[29] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.
[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1106–1114.
[32] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proc. CVPR*, 2014.
[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015.
[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, 2014.
[35] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *NIPS*, 2014.
[36] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2014.

TABLE 4
Comparison with state-of-the-art compact image representations (256-D) on image and object retrieval. We compare (b.) our best trained network, (a.) the corresponding off-the-shelf NetVLAD network (whitening learnt on Pittsburgh), and the state-of-the-art for compact image representations on standard image and object retrieval benchmarks. For completeness, we also include the Max pooling baselines (our best trained and off-the-shelf), reduced to 256-D with whitening. "original" and "rotated" for Holidays denote whether the original or the manually rotated dataset [41], [76] is used. The "crop" and "full" for Oxford/Paris correspond to the testing procedures when the query ROI is respected (the image is cropped as in [41]), or ignored (the full image is used as the query), respectively. † [44] use square patches whose side is equal to $1.5\times$ the maximal dimension of the query ROI (the detail is available in version 2 of their arXiv paper), so the setting is somewhere in between "crop" and "full", arguably closer to "full" as ROIs become very large.

Method	Oxford 5k		Oxford 105k	Paris 6k		Holidays	
	full	crop	crop	full	crop	original	rotated
Jégou and Zisserman [55]	–	47.2	40.8	–	–	65.7	65.7
Gordo et al. [106]	–	–	–	–	–	78.3	–
Razavian et al. [44]	53.3†	–	–	67.0†	–	74.2	–
Babenko and Lempitsky [41]	58.9	53.1	50.1	–	–	–	80.2
Ours: Max off-the-shelf	57.8	57.2	52.7	65.7	69.0	73.7	78.0
Ours: Max trained	**63.2**	**63.5**	59.8	**72.9**	72.8	74.2	78.1
a. Ours: NetVLAD off-the-shelf	53.4	55.5	52.2	64.3	67.7	**82.1**	**86.0**
b. Ours: NetVLAD trained	62.5	**63.5**	**60.8**	72.0	**73.5**	79.9	84.3

[37] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition," *CoRR*, vol. abs/1310.1531, 2013.

[38] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013.

[39] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. ECCV*, 2014.

[40] H. Azizpour, A. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability from a generic ConvNet representation," *CoRR*, vol. abs/1406.5774, 2014.

[41] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proc. ICCV*, 2015.

[42] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *Proc. ECCV*, 2014.

[43] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," *CoRR*, vol. abs/1403.6382, 2014.

[44] A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "A baseline for visual instance retrieval with deep convolutional networks," in *Proc. ICLR*, 2015.

[45] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations," in *Proc. ICLR*, 2016.

[46] Y. Kalantidis, C. Mellina, and S. Osindero, "Cross-dimensional weighting for aggregated deep convolutional features," in *ECCV workshop on Web-scale Vision and Social Media*, 2016.

[47] E. Mohedano, K. McGuinness, N. E. O'Connor, A. Salvador, F. Marqués, and X. Giró-i-Nieto, "Bags of local convolutional features for scalable instance search," in *ACM ICMR*, 2016.

[48] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *Proc. CVPR*, 2012.

[49] H. Jégou and O. Chum, "Negative evidences and co-occurrences in image retrieval: the benefit of PCA and whitening," in *Proc. ECCV*, 2012.

[50] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, "Total recall: Automatic query expansion with a generative feature model for object retrieval," in *Proc. ICCV*, 2007.

[51] O. Chum, A. Mikulik, M. Perďoch, and J. Matas, "Total recall II: Query expansion revisited," in *Proc. CVPR*, 2011.

[52] J. Delhumeau, P.-H. Gosselin, H. Jégou, and P. Pérez, "Revisiting the VLAD image representation," in *Proc. ACMM*, 2013.

[53] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Proc. ECCV*, 2008, pp. 304–317.

[54] ——, "On the burstiness of visual elements," in *Proc. CVPR*, Jun 2009.

[55] H. Jégou and A. Zisserman, "Triangulation embedding and democratic aggregation for image search," in *Proc. CVPR*, 2014.

[56] A. Mikulik, M. Perďoch, O. Chum, and J. Matas, "Learning a fine vocabulary," in *Proc. ECCV*, 2010.

[57] F. Perronnin and D. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proc. CVPR*, 2007.

[58] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *Proc. CVPR*, 2008.

[59] K. Simonyan, A. Vedaldi, and A. Zisserman, "Descriptor learning using convex optimisation," in *Proc. ECCV*, 2012.

[60] G. Tolias, Y. Avrithis, and H. Jégou, "To aggregate or not to aggregate: Selective match kernels for image search," in *Proc. ICCV*, 2013.

[61] G. Tolias and H. Jégou, "Visual query expansion with or without geometry: refining local descriptors by feature aggregation," *Pattern Recognition*, 2014.

[62] T. Turcot and D. G. Lowe, "Better matching with fewer features: The selection of useful features in large database recognition problems," in *ICCV Workshop on Emergent Issues in Large Amounts of Visual Data (WS-LAVD)*, 2009.

[63] H. Jégou, H. Harzallah, and C. Schmid, "A contextual dissimilarity measure for accurate and efficient image search," in *Proc. CVPR*, 2007.

[64] D. Qin, S. Gammeter, L. Bossard, T. Quack, and L. Van Gool, "Hello neighbor: accurate object retrieval with k-reciprocal nearest neighbors," in *Proc. CVPR*, 2011.

[65] D. Qin, C. Wengert, and L. V. Gool, "Query adaptive similarity for large scale object retrieval," in *Proc. CVPR*, 2013.

[66] J. Zepeda and P. Pérez, "Exemplar SVMs as visual feature encoders," in *Proc. CVPR*, 2015.

[67] D. Qin, Y. Chen, M. Guillaumin, and L. V. Gool, "Learning to rank bag-of-word histograms for large-scale object retrieval," in *Proc. BMVC.*, 2014.

[68] S. Winder, G. Hua, and M. Brown, "Picking the best DAISY," in *Proc. CVPR*, 2009, pp. 178–185.

[69] J. Philbin, M. Isard, J. Sivic, and A. Zisserman, "Descriptor learning for efficient retrieval," in *Proc. ECCV*, 2010.

[70] A. Makadia, "Feature tracking for wide-baseline image retrieval," in *Proc. ECCV*, 2010.

[71] A. Bergamo, S. N. Sinha, and L. Torresani, "Leveraging structure from motion to learn discriminative codebooks for scalable landmark classification," in *Proc. CVPR*, 2013.

[72] D. Qin, X. Chen, M. Guillaumin, and L. V. Gool, "Quantized kernel learning for feature matching," in *NIPS*, 2014.

[73] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, and F. Moreno-Noguer, "Fracking deep convolutional image descriptors," *CoRR*, vol. abs/1412.6537, 2014.

[74] M. Paulin, M. Douze, Z. Harchaoui, J. Mairal, F. Perronnin, and C. Schmid, "Local convolutional features with unsupervised training for image retrieval," in *Proc. ICCV*, 2015.

[75] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: Learned invariant feature transform," in *Proc. ECCV*, 2016.

[76] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *Proc. ECCV*, 2014.

[77] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in *Proc. ECCV*, 2016.

[78] F. Radenović, G. Tolias, and O. Chum, "CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples," in *Proc. ECCV*, 2016.

[79] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. CVPR*, 2016.

[80] T. Weyand, I. Kostrikov, and J. Philbin, "PlaNet – Photo geolocation with convolutional neural networks," in *Proc. ECCV*, 2016.

[81] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DOF camera relocalization," in *Proc. ICCV*, 2015.

[82] T.-Y. Lin, Y. Cui, S. Belongie, and J. Hays, "Learning deep representations for ground-to-aerial geolocalization," in *Proc. CVPR*, 2015.

[83] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *Proc. CVPR*, 2015.

[84] G. Csurka, C. Bray, C. Dance, and L. Fan, "Visual categorization with bags of keypoints," in *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[85] V. Sydorov, M. Sakurada, and C. Lampert, "Deep fisher kernels – end to end learning of the fisher kernel GMM parameters," in *Proc. CVPR*, 2014.

[86] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Fisher networks for large-scale image classification," in *NIPS*, 2013.

[87] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," in *Proc. ICCV*, 2015.

[88] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *NIPS*, 2004.

[89] K. Q. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *NIPS*, 2006.

[90] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proc. CVPR*, 2014.

[91] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. CVPR*, 2015.

[92] J. Foulds and E. Frank, "A review of multi-instance learning assumptions," *The Knowledge Engineering Review*, vol. 25, no. 01, pp. 1–25, 2010.

[93] D. Kotzias, M. Denil, P. Blunsom, and N. de Freitas, "Deep multi-instance transfer learning," *CoRR*, vol. abs/1411.3128, 2014. [Online]. Available: http://arxiv.org/abs/1411.3128

[94] P. Viola, J. C. Platt, and C. Zhang, "Multiple instance boosting for object detection," in *NIPS*, 2005.

[95] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE PAMI*, vol. 20, no. 1, pp. 23–38, Jan 1998.

[96] N. Dalal and B. Triggs, "Histogram of Oriented Gradients for Human Detection," in *Proc. CVPR*, vol. 2, 2005, pp. 886–893.

[97] P. F. Felzenszwalb, R. B. Grishick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE PAMI*, 2010.

[98] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[99] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[100] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. ICML*, 2016.

[101] P. Gronat, "`streetget` - a small python package for building place recognition datasets," http://www.di.ens.fr/willow/research/streetget/, 2015.

[102] "Project webpage: code, networks, data, results, presentation." [Online]. Available: http://www.di.ens.fr/willow/research/netvlad/

[103] A. Vedaldi and K. Lenc, "Matconvnet – convolutional neural networks for matlab," in *Proc. ACMM*, 2015.

[104] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, 2009.

[105] H. J. Kim, E. Dunn, and J.-M. Frahm, "Learned contextual feature reweighting for image geo-localization," in *Proc. CVPR*, 2017.

[106] A. Gordo, J. A. Rodríguez-Serrano, F. Perronnin, and E. Valveny, "Leveraging category-level labels for instance-level image retrieval," in *Proc. CVPR*, 2012.

[107] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. CVPR*, 2015.

Relja Arandjelović received the BA and MEng degrees from the University of Cambridge in 2009, and PhD from the University of Oxford in 2013. He then spent one year as a postdoctoral researcher at the University of Oxford, and two years as a postdoctoral researcher at INRIA / Ecole Normale Superieure, Paris. Since 2016, he is a senior research scientist at DeepMind. His research interests include large-scale visual search and place recognition.

Petr Gronat earned his Master degrees from the Czech Technical University in Prague and the Chung Hua University in Taiwan R.O.C. He then started his PhD at Czech Technical University in Prague and since 2012 he works as a research engineer at INRIA. His research interests include place recognition and autonomous driving.

Akihiko Torii received a Master degree and PhD from Chiba University in 2003 and 2006. He then spent four years as a post-doctoral researcher in Czech Technical University in Prague. Since 2010, he has been with Tokyo Institute of Technology, where he is currently an assitant professor in the Department of Systems and Control Engineering, the School of Engineering.

Tomas Pajdla received the MSc and PhD degrees from the Czech Technical University in Prague. He works in geometry and algebra of computer vision and robotics with emphasis on nonclassical cameras, 3D reconstruction, and industrial vision. He contributed to introducing epipolar geometry of panoramic cameras, noncentral camera models generated by linear mapping, generalized epipolar geometries, to developing solvers for minimal problems in structure from motion and to solving image matching problem. at OAGM 1998 and 2013, BMVC 2002 and ACCV 2014. He is a member of the IEEE. Google Scholar: http://scholar.google.com/citations?user=gnR4zf8AAAAJ

Josef Sivic received MSc degree from the Czech Technical University in Prague, PhD from the University of Oxford and Habilitation degree from Ecole Normale Superieure in Paris. He currently holds a permanent position as a senior researcher (directeur de recherche) at Inria in Paris. He has published more than 60 scientific publications, has served as an area chair for major computer vision conferences and as a program chair for ICCV'15. He currently serves as an associate editor for the International Journal of Computer Vision and IEEE Transactions on Pattern Analysis and Machine Intelligence. Since 2014, he leads ERC project LEAP.

Supplementary material for: NetVLAD: CNN architecture for weakly supervised place recognition

In this supplementary material, we provide additional discussions and results to complement the main paper.

Which layer to use for base descriptors? Our CNN architecture is based on aggregating activations produced by a particular CNN layer using NetVLAD. There are several choices regarding where exactly to perform the crop – which layer, before or after the ReLU, and whether to do some descriptor postprocessing, such as L2-normalization. Several works have already investigated these questions, including [1], [2], [3], [4], [5], [6], [7]. We have followed commonly used practice of cropping the networks at the last convolutional layer, and conducted preliminary experiments for the decision of using before- or after-ReLU activations, and whether to L2-normalize them or not. Our initial experiments with the off-the-shelf Max-pooling network showed that before-ReLU worked slightly better than after-ReLU, so we kept this setting throughout all experiments. We also done preliminary experiments with L2-normalization of the input descriptors, and observed that L2-normalization helped off-the-shelf NetVLAD slightly ($+2\%$), while substantially hurting off-the-shelf Max-pooling (-7%), so we kept the best setting for each of the methods, i.e. L2-normalization is done for NetVLAD inputs but not for Max-pooling.

Max and Sum with whitening. [2] suggest using whitening with Sum pooling, but this only makes sense for off-the-shelf networks – the whitening transformation can be incorporated inside the last conv layer due to the linearity of the transformation, so training a Sum pooling network is capable of learning the whitening implicitly. We therefore only test trained Max-pooling with whitening and observe a significant drop in performance on the Tokyo 24/7 benchmark (recall@5: from 52.7% to 42.2%).

Is pooling needed? Here we provide additional details to complement the discussion in the "is pooling needed" paragraph of the main paper. There are multiple issues with "no-pool" approaches. First, if no pooling is performed, then the image representation is not invariant to translation, scale or partial occlusions as it enforces a fixed spatial layout, and the visual search is akin to near-duplicate detection. An analogy with the pre-CNN image retrieval would be to use GIST [8] instead of bag-of-words or VLAD, which has been shown to be clearly inappropriate due to the aforementioned reasons [9].

Second, no-pool cannot handle variable image sizes, as, without pooling, the convolutional map varies in size depending on the input. This is a real problem as Tokyo 24/7 queries, and all image retrieval benchmarks (Oxford, Paris, Holidays) have variable image sizes. The problem can be handled by padding to a predefined fixed size (introducing boundary artifacts), resizing and sacrificing the original aspect ratio (can be very detrimental), or resizing isotropically and taking the central crop (loss of information outside of the crop).

Third, passing a 640×480 image through the VGG-16 network and extracting the activations at the last convolutional layer results in a $40 \times 30 \times 512$ feature map, i.e. the image representation would be 600k-D. Note that it is not simple to perform dimensionality reduction of such a large representation, e.g. to reduce the dimensionality to 512 using PCA one would need to estimate 315 million parameters, which is hard to do reliably. Therefore, it is necessary to down-sample images which results in reduced performance [6], [7].

To test no-pooling, 640×480 images are used as input since all database images have this resolution. Images that do not conform to this aspect ratio (Tokyo 24/7 queries) are resized isotropically to the smallest resolution that can accommodate a 640×480 rectangular region, and a central crop of this size is taken. The image is passed through a VGG-16 network cropped at conv5_3, but since this results in a huge $40 \times 30 \times 512$ feature map, local Max-pooling is applied to reduce it to a manageable size, comparable to NetVLAD. In more detail, Max-pooling is performed with kernel size and stride 5, resulting in a $8 \times 6 \times 512$ feature map. The feature map is vectorized followed by L2-normalization, and used as a 25k-D image representation. A fully connected layer (FC) is optionally appended to perform dimensionality reduction into 4096-D, followed by another L2-normalization. The no-pool and no-pool+FC representations are trained on TokyoTM using exactly the same procedure and loss as NetVLAD, and evaluated on Tokyo 24/7, making results directly comparable to Figure 5(c) of the main paper.

The no-pool+FC method overfits badly because the FC is very large ($25k \times 4096 = 100M$ parameters), achieving recall@5 of only 34.6%, while no-pool gets 63.5%, compared to 82.5% of NetVLAD%. No-pool does beat Max-pooling which gets 52.7%, but it does so with a 48 times much larger image representation (25k-D versus 512-D). Reducing no-pool down to 4096-D with whitening gives only 50.8%, i.e. a 8 times larger representation achieves a lower performance than Max-pooling.

Pretraining. A natural question is whether pretraining the convolutional layers by first training with Max-pooling can be beneficial for the subsequent NetVLAD training. To investigate this, an AlexNet based NetVLAD network was trained on Pitts30k-train

TABLE 1
Image and object retrieval for varying dimensionality of NetVLAD. We compare our best trained network (VGG-16, f_{VLAD}), and the corresponding off-the-shelf network (whitening learnt on Pittsburgh), on standard image and object retrieval benchmarks, while varying the dimensionality (Dim.) of the image representation.

Method	Dim.	Oxford 5k		Paris 6k		Holidays	
		full	crop	full	crop	orig	rot
NetVLAD off-shelf	16	28.7	28.1	36.8	38.2	56.6	60.3
	32	36.5	36.0	48.9	51.9	68.0	71.7
	64	40.1	38.9	55.7	58.1	76.5	80.4
	128	49.0	49.8	60.1	63.2	79.1	83.3
	256	53.4	55.5	64.3	67.7	82.1	86.0
	512	56.7	59.0	67.5	70.2	82.9	86.7
	1024	60.2	62.6	70.9	73.3	83.9	87.3
	2048	62.8	65.4	73.7	75.6	84.9	88.2
	4096	64.4	66.6	75.1	77.4	84.9	88.3
NetVLAD trained	16	32.5	29.9	45.1	44.9	54.8	58.6
	32	43.4	42.6	53.5	54.4	67.5	71.2
	64	53.6	51.1	61.8	63.0	75.4	79.3
	128	60.4	61.4	68.7	69.5	78.8	82.6
	256	62.5	63.5	72.0	73.5	79.9	84.3
	512	65.6	67.6	73.4	74.9	81.7	86.1
	1024	66.9	69.2	75.7	76.5	82.4	86.5
	2048	67.7	70.8	77.0	78.3	82.8	86.9
	4096	69.1	71.6	78.5	79.7	83.1	87.5

and tested on Pitts30k-val. We used the best Max-pooling network from Table 2 of the main paper (i.e. trained down to conv2), as the starting point for NetVLAD, and fine-tuned the NetVLAD network (also down to conv2). As expected, the Max-pooling pretraining learns better features, and therefore the initial performance of the NetVLAD representation, before any NetVLAD training, is much higher than for off-the-shelf NetVLAD: recall@5 on Pitts30k-val is 83.7% compared to 69.8%. However, the final performance after training is similar to that of the NetVLAD initialized with an off-the-shelf network, achieving recalls@1, 5 and 10 of 84.3%, 94.1% and 96.1%, respectively.

Qualitative evaluation. Figure 1 shows additional comparisons between our method and the best baseline (RootSIFT+VLAD+whitening).

Image retrieval. The complete set of NetVLAD results for different output dimensions is shown in table 1.

REFERENCES

[1] H. Azizpour, A. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability from a generic ConvNet representation," *CoRR*, vol. abs/1406.5774, 2014.

[2] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proc. ICCV*, 2015.

[3] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *Proc. ECCV*, 2014.

[4] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," *CoRR*, vol. abs/1403.6382, 2014.

[5] A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "A baseline for visual instance retrieval with deep convolutional networks," in *Proc. ICLR*, 2015.

[6] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations," in *Proc. ICLR*, 2016.

[7] Y. Kalantidis, C. Mellina, and S. Osindero, "Cross-dimensional weighting for aggregated deep convolutional features," in *ECCV workshop on Web-scale Vision and Social Media*, 2016.

[8] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *IJCV*, 2001.

[9] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid, "Evaluation of GIST descriptors for web-scale image search," in *Proc. CIVR*, 2009.

Fig. 1. **Examples of retrieval results for challenging queries on Tokyo 24/7.** Each column corresponds to one test case: the query is shown in the first row, the top retrieved image using our best method (trained VGG-16 NetVLAD + whitening) in the second, and the top retrieved image using the best baseline (RootSIFT + VLAD + whitening) in the last row. The green and red borders correspond to correct and incorrect retrievals, respectively. Note that our learnt descriptor can recognize the same place despite large changes in appearance due to illumination (day/night), viewpoint and partial occlusion by cars, trees and people. The last column corresponds to a difficult query, which is hard for our method because of its overall very dark appearance.