Fakultät für Mathematik, Informatik und Naturwissenschaften
Lehr- und Forschungsgebiet Informatik VIII
Computer Vision
Prof. Dr. Bastian Leibe

Seminar Report

Neural Codes for Image Retrieval

David Stutz
Matriculation number: ######

September 30, 2015

Advisor: Michael Kramp

Abstract

This seminar report focuses on using convolutional neural networks for image retrieval. Firstly, we give a thorough discussion of several state-of-the-art techniques in image retrieval by considering the associated subproblems: image description, descriptor compression, nearest-neighbor search and query expansion. We discuss both the aggregation of local descriptors using clustering and metric learning techniques as well as global descriptors.

Subsequently, we briefly introduce the basic concepts of deep convolutional neural networks, focusing on the architecture proposed by Krizhevsky et al. [KSH12]. We discuss different types of layers commonly used in recent architectures, for example convolutional layers, non-linearity and rectification layers, pooling layers as well as local contrast normalization layers. Finally, we shortly review supervised training techniques based on stochastic gradient descent and regularization techniques such as dropout and weight decay.

Finally, following Babenko et al. [BSCL14], we discuss the use of feature activations in intermediate layers as image representation for image retrieval. After presenting experiments and comparing convolutional neural networks for image retrieval with other state-of-the-art techniques, we conclude by motivating the combined use of deep architectures and hand-crafted image representations for accurate and efficient image retrieval.

Contents

1 Introduction

Content-based image or object retrieval, the problem of finding images within a large database containing the same or similar objects or scenes as a given query image, is a fundamental problem in computer vision. In particular, in order to extract useful information from large databases of images these databases need to be organized and searchable. Originally, images were manually annotated with keywords and text-based retrieval systems were utilized. However, due to the rapidly increasing size of image collections, manual annotation became infeasible. Therefore, content-based retrieval systems relying on image content only were developed and are heavily researched within the computer vision community. In the following, we always speak of content-based image retrieval and include the subproblem of object retrieval implicitly (in practice, techniques can often be applied to both problems [PCI+07]).
While the problem has been studied for several decades, recent success is based on the Bag of Visual Words model proposed by Sivic and Zisserman [SZ03]. Local descriptors, used to characterize interest points within the image, are quantized into so-called visual words and the image is represented by the corresponding word counts. While the Bag of Visual Words model has steadily been improved (for example [CPS+07, PCI+07, AZ12]), different techniques of aggregating local descriptors (for example [JDCSP10, GKS13, JZ14, PISZ10]) have been used and global descriptors have been used, as well (for example [OT01]).
Similar to image retrieval, convolutional neural networks [LBD+89] have been studied for more than two decades (see [Ben09, LBD+89]) but had their breakthrough only recently [Ben09]. In particular Krizhevsky et al. [KSH12] demonstrated astounding results on the ImageNet classification challenge [1]. Thus, convolutional neural networks have been applied to a range of different applications, and the usefulness of the generated intermediate activations has been studied within the literature [Ben09]. Recently, Babenko et al. [BSCL14] used convolutional neural networks, in particular the architecture proposed by Krizhevsky et al., for image retrieval.

Outline. This report is intended to introduce the reader to recent techniques in image retrieval. Furthermore, we briefly introduce the fundamentals of convolutional neural networks in order to discuss their application to image retrieval. In Section 2 we introduce the problem of image retrieval and discuss aggregation techniques for local descriptors as well as the application of global descriptors. Furthermore, we briefly introduce approximate nearest neighbor techniques, discriminative dimensionality reduction and average query expansion. In Section 3 we discuss the mathematical foundation of convolutional neural networks including their training and regularization. Subsequently, in Section 4, we discuss the application of convolutional neural networks for image retrieval before presenting experimental results by Babenko et al. Finally, we conclude in Section 6.

[1] See `http://www.image-net.org/challenges/LSVRC/2010/`.

2 Image Retrieval

Given a database of images $X = \{x_1, \ldots, x_N\}$ and a query image z_0 showing a particular object or scene, the task of image retrieval is usually formalized as follows:

Problem. Find an ordered list of images $Z = (z_1, \ldots, z_K)$ such that, for an appropriate distance function d, it holds:

$$d(z_0, z_k) \leq d(z_0, z_{k+1}), 1 \leq k < K \quad \text{and} \quad d(z_0, z_k) < d(z_0, x_n) \forall 1 \leq k \leq K, x_n \notin Z \tag{1}$$

which basically describes a K-nearest neighbor search.

While early work on image retrieval presented complete retrieval systems, recent publications usually focus on subproblems. Therefore, in the following, we roughly divide this problem into image description, compression, nearest neighbor search and query expansion.

2.1 Image Description

A fundamental problem in image retrieval is image representation, that is the characterization of an image using a feature vector of fixed dimensionality. We distinguish global descriptors directly constituting an image representation as well as local descriptors which usually need to be aggregated into an image representation of fixed dimensionality.

2.1.1 Aggregation of Local Descriptors

Local descriptors are designed to represent regions around interest points. For image retrieval, these interest points are computed using a scale, rotation and affine invariant detector. We refer to [MS04] for details and discuss the Scale-Adapted Harris Detector as example [MS04].

Scale-Adapted Harris Detector [MS04, HS88]. The Harris detector is based on the second moment matrix A of an image x_n (see Equation (2)). The corresponding eigenvalues λ_1, λ_2 represent the signal change in two orthogonal directions and interest points are extracted at pixels where both eigenvalues are large. For efficiency, Harris and Stephens [HS88] propose to maximize

$$\lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = \det(A) - \kappa\text{trace}(A)^2 \quad \text{with} \quad A = \begin{pmatrix} \partial_x^2 x_n & \partial_{xy} x_n \\ \partial_{xy} x_n & \partial_y^2 x_n \end{pmatrix} \tag{2}$$

where κ is a sensitivity parameter. In practice, the detector is applied in scale space, that is on a set of images

$$x_n^{(\sigma_s)} = g_{\sigma_s} * x_n \tag{3}$$

where $*$ denotes convolution and σ_s is sampled at logarithmic scale. Therefore, it automatically selects the optimal scale and defines the size of the region used for local descriptors. Several extensions have been proposed, for example the Harris-Laplace Detector uses the Scale-Adapted Harris Detector to detect interest points in a coarse scale space, and then refines detections by searching for extrema of the Laplacian-of-Gaussians

$$\Delta(g_{\sigma_s} * x_n) = \partial_{xx}(g_{\sigma_s} * x_n) + \partial_{yy}(g_{\sigma_s} * x_n), \tag{4}$$

which detects blob-like structures, in a finer scale space.

Note that instead of computing interest points, local descriptors can also be computed densily. It has been shown that this approach may reduce runtime and increase performance [GRPV12].

Local Descriptors

Given a set of interest points, local descriptors are intended to compute discriminative characterizations of the corresponding regions. We follow [Low04] and briefly introduce the most commonly used local descriptor, **SIFT**, and a simplistic local descriptor used for image retrieval in [GKS13].

Scale Invariant Feature Transform (SIFT) [Low04]. First, given a specific scale, the orientation of the interest point is determined by computing a gradient orientation histogram under a Gaussian window and using the maximum bin as orientation. In practice, 36 bins are used to determine the orientation and up to three orientations corresponding to bins above 80% of the maximum bin are retained as additional orientations. Then, **SIFT** divides a rectangular region, rotated according to the orientation determined previously, around the interest point into a 4×4 grid. For each grid element, a 8-bin gradient orientation histogram is computed using trilinear interpolation. Pixels are weighted according to gradient magnitude and a Gaussian window. This results in a $c = 4 \cdot 4 \cdot 8$-dimensional descriptors which is L_2-normalized.
In image retrieval, **SIFT** is the most commonly used local descriptor and several aggregation techniques are tailored specificly to **SIFT** descriptors (for example [JDCSP10]). Nevertheless, several extensions have been proposed, especially concerning normalization. For example, Arandjelović and Zisserman [AZ12] propose to use **RootSIFT**, that is the original **SIFT** descriptor is L_1-normalized and the individual elements are square-rooted. As result, the Euclidean distance on **RootSIFT** resembles the Bhattacharyya coefficient of the original **SIFT** descriptors.

Sparse-Coded Micro Feature [GKS13]. In their sparse-coding and max pooling framework (see next paragraph), Ge et al. use a simplistic local descriptor consisting only of the color values within the region around the interest point. For example, for a 5×5 region, the descriptor has dimension $c = 3 \cdot 5^2$. Ge et al. argue that after sparse-coding and max pooling, meaningful descriptors are picked and small patches are invariant to affine transformations.

Embedding and Aggregation Techniques

To obtain a global representation of an image, local descriptors are aggregated. Following recent publications as [GKS13] or [JZ14], aggregation techniques are divided into an embedding step and the actual aggregation step (also referred to as encoding and pooling in [GKS13]). Given a set of extracted descriptors $Y_n = \{y_{1,n}, \ldots, y_{L,n}\} \subset \mathbb{R}^c$ from image x_n, the goal is the computation of a C-dimensional image representation.

Bag of Visual Words [SZ03]. Sivic and Zisserman, motivated by early text-retrieval systems, cluster all local descriptors $Y = \bigcup_{n=1}^{N} Y_n$ using k-means clustering to define a vocabulary of visual words $\hat{Y} = \{\hat{y}_1, \ldots, \hat{y}_M\}$. Subsequently, descriptors are assigned to the nearest visual word and the global image representation is a sparse vector of word counts. Thus, the embedding step computes

$$f(y_{l,n}) = (\delta(NN_{\hat{Y}}(y_{l,n}) = \hat{y}_1), \ldots, \delta(NN_{\hat{Y}}(y_{l,n}) = \hat{y}_M)) \tag{5}$$

where $NN_{\hat{Y}}(y_{l,n})$ denotes the nearest neighbor of $y_{l,n}$ in $\hat{Y}$, that is $f_m(y_{l,n}) = 1$ if and only if $NN_{\hat{Y}}(y_{l,n}) = \hat{y}_m$. The aggregation step can be formalized as

$$F(Y_n) = \sum_{l=1}^{L} f(y_{l,n}). \tag{6}$$

In practice, however, the so called term-frequency inverse-document-frequency weighting [SZ03] is applied:

$$F_m(Y_n) = \frac{\sum_{l=1}^{L} f_m(y_{l,n})}{\sum_{m'=1}^{M} \sum_{l=1}^{L} f_{m'}(y_{l,n})} \log\left(\frac{N}{\sum_{n=1}^{N} \sum_{l=1}^{L} f_m(y_{l,n})}\right) \tag{7}$$

where $f_m(y_{l,n})$ and $F_m(Y_n)$ denote component m of the corresponding embedding and aggregation step, respectively. The first term is the fraction of local descriptors assigned to visual word $\hat{y}_m$ and, thus, determines the importance of $\hat{y}_m$ in the image representation of image x_n. In contrast, the second term down-weights the

influence of local descriptors assigned to word $\hat{y}_m$ if it occurs frequently in the whole database. To alleviate the influence of bursty elements, that is few large elements in the image representation [AZ13], $F(Y_n)$ can alternatively be square-rooted element-wise and re-normalized. The final image representation has $C = M$ dimensions.
An early problem with this approach is of computional nature. In particular, it may be infeasible to apply k-means clustering on Y for large databases (for example Philbin et al. [PCI$^+$07] argue that even subsampling the descriptors of a $100k$ database would result in millions of local descriptors). While Nistér and Steénius [NS06] proposed to use hierarchical k-means clustering, Philbin et al. [PCI$^+$07] introduced an approximation. Hierarchical k-means proceeds by iteratively re-clustering a set of M^* initial clusters – that is, a clustering-tree with branching factor M^* and depth Q is generated, resulting in $(M^*)^Q$ visual words. Approximate k-means clustering replaces the exact distance computation by an approximation based on randomized k-d trees [2].

Fisher Vectors [PD07, JH99]. Following [PD07] and [JH99], we first introduce the mathematical background of Fisher Vectors before making embedding and aggregation explicit. Therefore, we consider a Gaussian mixture model

$$p(y_{l,n}) = \sum_{m=1}^{M} w_m \mathcal{N}(y_{l,n}|\mu_m, \Sigma_m), \quad \sum_{m=1}^{M} w_m = 1 \tag{8}$$

where $\mathcal{N}(y_{l,n}|\mu_m, \Sigma_m)$ refers to a Gaussian with mean μ_m and covariance Σ_m. The model is usually learned on Y using the Maximum Likelihood criterion by employing a standard Expectation Maximization algorithm. The idea of Fisher vectors is to characterize a local descriptor $y_{l,n}$ by the following gradient:

$$\nabla_{\mu_m} \log(p(y_{l,n})). \tag{9}$$

Intuitively, Equation (9) characterizes each descriptor $y_{l,n}$ by the direction in which the descriptor should be adapted to better fit the Gaussian mixture model. Taking into account all local descriptors Y_n of image x_n, which are assumed to be independent, the log-likelihood can be written as

$$\log(p(Y_n)) = \sum_{l=1}^{L} \log\left(p(y_{l,n})\right). \tag{10}$$

The partial derivative of the log-likelihood with respect to the mean μ_m is given as

$$\sum_{l=1}^{L} \gamma_m(y_{l,n}) \Sigma_m^{-1} (y_{l,n} - \mu_m) \tag{11}$$

where $\gamma_m(y_{l,n})$ describes the probability of descriptor $y_{l,n}$ belonging to component m:

$$\gamma_m(y_{l,n}) = \frac{w_m \mathcal{N}(y_{l,n}|\mu_m, \Sigma_m)}{\sum_{m'=1}^{M} w_{m'} \mathcal{N}(y_{l,n}|\mu_{m'}, \Sigma_{m'})} \tag{12}$$

In practice, the covariance matrix Σ_m is assumed to be diagonal, that is $\Sigma_m = \mathrm{diag}(\sigma_{1,m}^2, \ldots, \sigma_{c,m}^2)$. Further, the gradient vectors are normalized using the Fisher information matrix

$$Z = \mathbb{E}_Y \left[\nabla \log(p(Y_n)) \nabla \log(p(Y_n))^T\right] \tag{13}$$

for which Perronnin et al. derive the following approximation:

$$Z_{\mu_m}^{-1} \nabla_{\mu_m} \log(p(y_{l,n})) \quad \text{with} \quad Z_{\mu_m} = \frac{L w_m}{\sigma_m^2}. \tag{14}$$

Here, the inversion of Z_{μ_m} as well as the division by σ_m^2 is meant element-wise.
Based on the above derivation, we use

$$f(y_{l,n}) = \left(Z_{\mu_1}^{-1} \nabla_{\mu_1} p(y_{l,n}), \ldots, Z_{\mu_M}^{-1} \nabla_{\mu_M} p(y_{l,n})\right) \tag{15}$$

[2] An implementation is available at `http://www.robots.ox.ac.uk/~vgg/software/fastanncluster/`.

as the embedding step and Equation (6) for aggregation. The image representation has a dimensionality of $C = Mc$ and is usually power-law normalized, that is

$$F_m(Y_n) = \text{sign}\left(F_m(Y_n)\right) |F_m(Y_n)|^\alpha \tag{16}$$

with $\alpha \in (0,1)$. Commonly, α is chosen as $\alpha = 0.5$ and, similar to **RootSIFT**, the normalization is then referred to as signed square-rooting. Subsequently, the image representation is L_2-normalized. Typical values are $M = 64$ or $M = 256$ and $c = 128$ for **SIFT** descriptors.

Vector of Locally Aggregated Descriptors (VLAD) and **Vocabulary Adaptation** [JDCSP10, AZ13]. Here, similar to the Bag of Visual Words model, the embedding step is guided by a vocabulary of M visual words learned using k-means clustering. Instead of counting word occurrences, Jégou et al. [JDCSP10] consider the corresponding residuals:

$$f(y_{l,n}) = (\delta(NN_{\hat{Y}}(y_{l,n}) = \hat{y}_1)(y_{l,n} - \hat{y}_1), \ldots, \delta(NN_{\hat{Y}}(y_{l,n}) = \hat{y_M})(y_{l,n} - \hat{y}_M)) \,. \tag{17}$$

The corresponding aggregation step is given by Equation (6). The image representation is subsequently either L_2-normalized or power-law normalized [AZ13]. In contrast, the sum of residuals of each visual word can also be L_2-normalized independently before L_2-normalizing the whole image representation [AZ13]. This normalization scheme is termed intra-normalization. The final image representation has dimension $C = Mc$.

In [AZ13], Arandjelovic and Zisserman additionally propose Vocabulary Adaptation. As discussed earlier, obtaining a vocabulary of visual words using k-means clustering may be computationally expensive for large databases. In addition, the visual words would have to be re-computed if the database is growing over time. Vocabulary Adaptation allows to adapt **VLAD** descriptors computed over an arbitrary database to an updated – or even new – database and proceeds as follows. First each visual word is adapted to be the mean of local descriptors assigned to it. Note that this is not equivalent to re-clustering the set of all local descriptors as the assignments of local descriptors to visual words do not change. Then, the **VLAD** descriptors are re-computed taking into account the adapted visual words. It is important to note that this second step does not need to access the original local descriptors.

Sparse-Coded Features [GKS13]. Given a vocabulary of visual words, Ge et al. use sparse codes as embedding:

$$f(y_{l,n}) = \arg\min_{r_l} \|y_{l,n} - \hat{Y} r_l\|_2^2 + \lambda \|r_l\|_1 \tag{18}$$

where $\hat{Y}$ contains the visual words $\hat{y}_m$ as columns and λ is a regularization parameter. This can be interpreted as linear regression with L_1 regularization. As second step, these sparse codes are pooled into a single M-dimensional feature vector. Max pooling is given by

$$F(Y_n) = \left(\max_{1 \le l \le L} \{f_1(y_{l,n})\}, \ldots, \max_{1 \le l \le L} \{f_M(y_{l,n})\} \right), \tag{19}$$

while Ge et al. refer to Equation (6) as average pooling. The final image representation is L_2-normalized and has dimension $C = M$.

Ge et al. show that their approach is able to aggregate different types of local descriptors. In particular, after extracting K' sets of local descriptors $Y_n^{(1)}, \ldots, Y_n^{(K')}$, the above aggregation technique is applied on each set individually and the obtained representations are concatenated:

$$f(y_{l,n}^{(k)}) = \left(0, \ldots, r_{l,1}^{(k)}, \ldots, r_{l,M}^{(k)}, \ldots, 0\right) \tag{20}$$

where $r_{l,m}^{(k)}$ is defined as above after minimizing Equation (**??**) on $Y_n^{(k)}$ for pre-computed visual words $\hat{y}_1^{(k)}, \ldots, \hat{y}_M^{(k)}$ (assuming the vocabulary size for the different types of local descriptors to be M). Max pooling and average pooling are applied as in Equation (19) and Equation (6), respectively, on the concatenated image representation.

Triangulation Embedding and Democratic Aggregation [JZ14]. Jégou and Zisserman are motivated by the aggregation function used in the Bag of Visual Words model or **VLAD**, see Equation (6), which gives unequal weight to the descriptors in Y_n. We follow [JZ14] and motivate their approach by considering the set similarity

$$F(Y_n)^T F(Y_n) = \sum_{l=1}^{L} \sum_{l'=1}^{L} f(y_{l,n})^T f(y_{l',n}). \tag{21}$$

Note that Equation (21) can also be written using a kernel $k(y_{l,n}, y_{l',n})$ as done in [JZ14]. Further, we investigate the contribution of a single descriptor $y_{l,n}$ to the set similarity in Equation (21):

$$f(y_{l,n})^T \sum_{l'=1}^{L} f(y_{l',n}) = \|f(y_{l,n})\|_2^2 + f(y_{l,n})^T \sum_{l' \neq l} f(y_{l',n}). \tag{22}$$

Jégou and Zisserman approach the second term at the right hand side of Equation (22) by minimizing the inner product of unrelated local descriptors. Instead of using the absolute distance between local descriptors and visual words, the Triangulation Embedding uses directional information only. Therefore, we consider the normalized residual

$$r_{l,m} = \frac{y_{l,n} - \hat{y}_m}{\|y_{l,n} - \hat{y}_m\|_2}. \tag{23}$$

Let $r_l = (r_{l,1}^T, \ldots, r_{l,M}^T)$ be the concatenation of these residuals. Then, the embedding step is given as

$$f(y_{l,n}) = (\Sigma(r_l))^{-\frac{1}{2}} (r_l - \mu(r_l)) \tag{24}$$

which essentially is a whitening operation, that is $\Sigma(r_l)$ and $\mu(r_l)$ are covariance and mean estimated on $\bigcup_{n=1}^{N} \{r_l\}_{l=1}^{L}$. Further, Jégou and Zisserman experimentally show that removing the components corresponding to the largest eigenvalues, which is equivalent to removing the first components of $f(y_{l,n})$, reduces the similarity between unrelated descriptors. Finally, $f(y_{l,n})$ is usually L_2-normalized.
Considering the set similarity of Equation (21), all local descriptors should contribute equally during aggregation. Therefore, Jégou and Zisserman introduce the concept of a democratic kernel: A kernel $k(y_{l,n}, y_{l',n})$, for example $k(y_{l,n}, y_{l',n}) = y_{l,n}^T y_{l',n}$, is said to be democratic if there exists a constant κ (which may or may not depend on the set Y) such that

$$\sum_{l'=1}^{L} k(y_{l,n}, y_{l',n}) = \kappa \quad \forall 1 \leq l \leq L. \tag{25}$$

Intuitively, this constraint demands that all local descriptors contribute equally to the set similarity. To accomplish this, a kernel can be replaced by a weighted counterpart:

$$k'(y_{l,n}, y_{l',n}) = \lambda_l \lambda_{l'} k(y_{l,n}, y_{l',n}) \tag{26}$$

The weights are learned by solving

$$\lambda_l \left(\sum_{l'=1}^{L} \lambda_{l'} k(y_{l,n}, y_{l',n}) \right) = \kappa \quad \text{with} \quad \lambda_l > 0, \forall 1 \leq l \leq L. \tag{27}$$

using an adapted Sinkhorn algorithm [3] [Sin64]. Based on the computed weights, the aggregation step is given as

$$F(Y_n) = \sum_{l=1}^{L} \lambda_l f(y_{l,n}) \tag{28}$$

[3] In 1964, Sinkhorn [Sin64] originally approached the problem of generating a doubly stochastic matrix S, that is a matrix satisfying $\sum_i s_{i,j} = 1$ and $\sum_j s_{i,j} = 1$, $\forall i, j$. Therefore, given an arbitrary matrix A, the problem of finding diagonal matrices Λ_1 and Λ_2 such that $S = \Lambda_1 A \Lambda_2$ is doubly stochastic is equivalent to solving Equation (27) with $\kappa = 1$. Sinkhorn proofs that for positive A (that is $a_{i,j} > 0$) such matrices Λ_1, Λ_2 exist and iteratively normalizing the rows and the columns of A converges to S. Further details can also be found in the appendix of [JZ14].

The final image representation has dimensionality $C = Mc$ and is both power-law normalized and L_2-normalized.

Descriptor Learning for Image Retrieval [PISZ10]. Philbin et al. try to avoid quantization errors, often incurred when obtaining visual words using k-means clustering, by learning a linear transformation $P \in \mathbb{R}^{c' \times c}$ such that matching descriptors lie more closely together and non-matching descriptors farther apart. First, a dataset of descriptor pairs is generated as follows: We randomly select a pair of images, detect interest points and extract local descriptors. Then, we apply Lowe's (second-nearest neighbor) ratio test [Low04]: To filter out local descriptors without correct match within the database (for example, due to background variation), descriptors where the fraction of distances to the first and the second nearest-neighbor is above 0.8 are discarded. Here, the second nearest-neighbor is constrained to come from an unrelated image. Finally, we estimate an affine transformation using **RANSAC** . The dataset includes the inliers found by **RANSAC** as positives $Y_{\text{pos}} \subset \mathbb{R}^c \times \mathbb{R}^c$, the outliers found by **RANSAC** as hard negatives Y_{out} and random matches as easy negatives Y_{rnd}. Philbin et al. minimize the following objective:

$$\begin{aligned} E(P) = &\sum_{(y_{l,n}, y_{l',n}) \in Y_{\text{pos}}} \mathcal{L}(b_1 - d(Py_{l,n}, Py_{l',n})) + \sum_{(y_{l,n}, y_{l',n}) \in Y_{\text{out}}} \mathcal{L}(d(Py_{l,n}, Py_{l',n}) - b_1) \\ &+ \sum_{(y_{l,n}, y_{l',n}) \in Y_{\text{rnd}}} \mathcal{L}(d(Py_{l,n}, Py_{l',n}) - b_2) + \frac{\lambda}{2}\|P\|^2. \end{aligned} \tag{29}$$

Intuitively, this is a maximum margin approach where λ controls the regularization term and

$$\mathcal{L}(z) = \log(1 + \exp(-z)) \tag{30}$$

is the logistic loss. Equation (29) is minimized using stochastic gradient descent, see Section 3.5. In practice, the whole procedure is applied on **SIFT** [Low04] descriptors and subsequently the Bag of Visual Words model is used. The final image representation has dimension $C = Mc'$.

2.1.2 Global Descriptors

Global descriptors are not based on interest points but directly compute a global image representation. Here, we intend to discuss the **GIST** [OT01] descriptor as example of a global descriptor used in image retrieval.

GIST [OT01]. In [OT01], Oliva and Torralba propose the Spatial Envelope representation which is used interchangeably with **GIST** and is intended to model the following scene properties: (i) naturalness is intended to distinguish natural from human-made scenes; (ii) openness refers to the difference between open scenes and enclosed scenes; (iii) roughness refers to the size of the parts of a scene; (iv) expansion refers to the depth perception of a scene; (v) and ruggedness characterizes to what extent the horizontal line of a scene is visible. We refer to [OT01] for details and instead focus on the actual computation of **GIST** [4]. In a first step, the image is down-scaled and a blurred version of the image is subtracted. The result is contrast normalized. Then, a set of Gabor-like filters at multiple scales and with multiple orientations are applied. The image is subdivided into a grid of rectangles and average pooling is applied. Finally, the responses in each rectangle, over all scales and orientations, are combined into a single global descriptor. Typically, 8 orientations at 4 scales are used and the image is divided into a 4×4 grid resulting in an $C = 8 \cdot 4 \cdot 16$-dimensional descriptor.

2.2 Descriptor Compression, Indexing and Nearest Neighbor Search

Given image representations [5] $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^C$ of all images within the database, we are interested in the nearest neighbors of the query z_0 in X [JDS11]. As solving Equation (1) may be infeasible for large databases, authors often discuss compression of their proposed image representation. While **PCA** is frequently used within the image retrieval community (for example [BSCL14, GKS13, AZ13]), discriminative techniques may be beneficial. As example, we discuss the approach by Gordo et al. [GRPV12].

[4] We follow the original implementation of **GIST** available at `http://people.csail.mit.edu/torralba/code/spatialenvelope/`.

[5] In the following, we abuse notation and use x_n to refer to both the image and the image representation.

Joint Subspace and Classifier Learning [GRPV12]. Given a labeled training set

$$\{(x_1,t_1),\ldots,(x_N,t_N)\},\quad t_n \in \{1,\ldots,T\}, \tag{31}$$

Joint Subspace and Classifier Learning aims to project the image representations and their labels into a common subspace where relevant labels are closer to the corresponding image representation than irrelevant ones. Gordo et al. propose to jointly learn a dimensionality reduction $P \in \mathbb{R}^{C' \times C}$ and a set of classifiers $w_t \in \mathbb{R}^{C'}, t \in \{1,\ldots,T\}$, in a large-margin framework. Therefore, the relevance of label t to image representation x_n is expressed as

$$s(x_n,t) = (Px_n)^T w_t. \tag{32}$$

Then,

$$E(P) = \sum_{(x_n,t_n,t):t\neq t_n} \max\{0, 1 - s(x_n,t_n) + s(x_n,t)\} \tag{33}$$

is minimized using stochastic gradient descent. In particular, after sampling a triple (x_n,t_n,t) with $t \neq t_n$, the projection matrix P and the classifiers w_{t_n} and w_t are updated only if the loss

$$\max\{0, 1 - s(x_n,t_n) + s(x_n,t)\} \tag{34}$$

is positive. Then the corresponding updates in iteration $[\tau+1]$ are given as

$$P[\tau+1] = P[\tau] + \gamma(w_{t_n}[\tau] - w_t[\tau])x_n^T \tag{35}$$

$$w_{t_n}[\tau+1] = w_{t_n}[\tau] + \gamma P[\tau]x_n \tag{36}$$

$$w_t[\tau+1] = w_t[\tau] - \gamma P[\tau]x_n \tag{37}$$

where γ is the learning rate, see Section 3.5. While P is subsequently used for dimensionality reduction, the learned classifiers w_t are discarded.

Now, given a (possibly low-dimensional) set of image representations, Equation (1) can be solved using exhaustive nearest neighbor search with complexity $O(NC)$. In contrast, approximate nearest neighbor search (see [JDS11] for references) tries to reduce complexity and memory consumption. As example, we discuss the approach proposed by Jégou et al. [JDS11].

Product Quantization [JDS11]. In general, a quantizer q is a function mapping each image representation to one of M centroids. Essentially, a quantizer tries to reconstruct a given database by M representatives and, thus, the reconstruction error can be expressed as

$$MSE(q) = \int p(x_n) d(x_n, q(x_n))^2 dx_n \tag{38}$$

$$\approx \sum_{n=1}^{N} d(x_n, q(x_n))^2. \tag{39}$$

Product Quantization subdivides each image representation into subvectors and each subvector is quantized separately using k-means clustering. The advantage of the product quantization lies in reduced memory consumption. In particular, k-means clustering requires to store $O(MC)$ floating point values, while Product Quantization stores $O(QM^*C) = O(M^{\frac{1}{Q}}C)$ with Q being the number of quantizers and $M = (M^*)^Q$.
Based on the above quantization, Jégou et al. propose two approaches to approximate search within the quantized image representations. Using symmetric distance computation, the distance $d(x_n,z_0)$ is approximated by

$$d(x_n,z_0) \approx \hat{d}(x_n,z_0) = d(q(x_n),q(z_0)). \tag{40}$$

In contrast, asymmetric distance computation is given as

$$\hat{d}(x_n,z_0) = d(x_n,q(z_0)). \tag{41}$$

Jégou et al. provide guarantees on the error of these approximations: the distance error of symmetric distance computation is statistically bounded by $MSE(q)$, while the distance error of asymmetric distance computation is statistically bounded by $2MSE(q)$. In practice, both bounds can be computed using Equation (39).
The above approach is still based on exhaustive search. Jégou et al. use an inverted filesystem to circumvent exhaustive search. Before using product quantization, the image representations are first quantized into a coarse quantization (with the number of centroids being significantly smaller compared to the product quantization). Each image representation is stored in a list assigned to the corresponding centroid of the coarse quantization. Instead of exhaustive search, only one of these lists is searched for nearest neighbors. We refer to [JDS11] for details.

2.3 Query Expansion

After retrieving a list of nearest neighbors, several techniques can be employed to improve query results: query expansion (for example [AZ12, CPS+07]) and spatial verification (for example [PCI+07]). As example, we briefly discuss average query expansion [CPS+07], however, refer to [PCI+07] for details on spatial verification.

Average Query Expansion [CPS+07]. Query expansion aims to improve query results by re-issuing a number of highly ranked results as new query to include further relevant results. Average Query Expansion uses the top K^* results and averages the corresponding term-frequency [6] representation:

$$z_{\text{avg}} = \frac{1}{K^*+1}\left(z_0 + \sum_{k=1}^{K^*} z_k\right). \tag{42}$$

The results of query z_{avg} is appended to the first K^* results of the original query. Usually, this type of query expansion is used in combination with spatial verification such that only verified results are included in the query expansion.

[6] Note that Chum et al. [CPS+07] use the Bag of Visual Words model with term-frequency weighting only (that is, only the first term in Equation (7)).

3 Convolutional Neural Networks

In 1989, LeCun et al. [LBD+89] underline the benefit if incorporating prior knowledge into neural networks. As result, convolutional neural networks are able to make use of the spatial information between pixels by using discrete convolution on images as raw input. After introducing different types of layers – the main building blocks of convolutional neural networks –, we focus on the architecture introduced by Krizhevsky et al. [KSH12] which is used by Babenko et al. [BSCL14] for image retrieval. First, however, we introduce the basic idea of neural networks using the multilayer-perceptron.

3.1 Neural Networks

The prototypical model of neural networks is the L-layer perceptron. Given the output $y^{(l-1)} \in \mathbb{R}^{m^{(l-1)}}$ of layer $(l-1)$, layer l computes

$$y_i^{(l)} = f(z^{(l)}) = f\left(\sum_{j=1}^{m^{(l-1)}} w_{i,j}^{(l)} y_j^{(l-1)} + w_{0,i}^{(l)}\right), \quad 1 \leq i \leq m^{(l)} \tag{43}$$

where f is an activation function which is applied component-wise [Bis95]. Common activations functions are the logistic sigmoid, given by

$$f(z) = \frac{1}{1+\exp(-z)}, \tag{44}$$

and the hyperbolic tangent. For an input vector $y^{(0)} := x$, a L-layer perceptron represents a parameterized model where the weights $w_i^{(l)}$ may be adapted to approximate a particular function. For classification, layer L is expected to model posterior probabilities which can be accomplished using the softmax activation function:

$$y_i^{(L)} = \frac{\exp\left(z_i^{(L)}\right)}{\sum_{j=1}^{m^{(L)}} \exp\left(z_j^{(L)}\right)}. \tag{45}$$

3.2 Layer Types

Similar to L-layer perceptrons, convolutional neural networks can be broken down into L layers. Different layer types are used to allow raw images as input, incorporate invariance to noise and distortions and accelerate training. We follow [JKRL09] to introduces these layer types.

3.2.1 Convolutional Layer

The convolutional layer is the key-ingredient of a convolutional neural network as it allows to handle multi-channel images as raw input. If layer l is a convolutional layer, its input is given by $m_1^{(l-1)}$ feature maps $Y_j^{(l-1)}$ from the previous layer, each of size $m_2^{(l-1)} \times m_3^{(l-1)}$. Then, layer l computes $m_1^{(l)}$ feature maps as

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} W_{i,j}^{(l)} * Y_j^{(l-1)}, \quad \forall 1 \leq i \leq m_1^{(l)}, \tag{46}$$

where $B_i^{(l)}$ is a matrix of biases and $W_{i,j}^{(l)}$ is a matrix of weights used as discrete filter. The size $m_2^{(l)} \times m_3^{(l)}$ of the feature maps $Y_i^{(l)}$ is dependent on the filter size as well as border effects. For $l = 1$, referring to the input image channels as $Y_1^{(0)}, \ldots, Y_3^{(0)}$, the layer directly operates on the input image. The convolutional layer is illustrated in Figure 1(a).

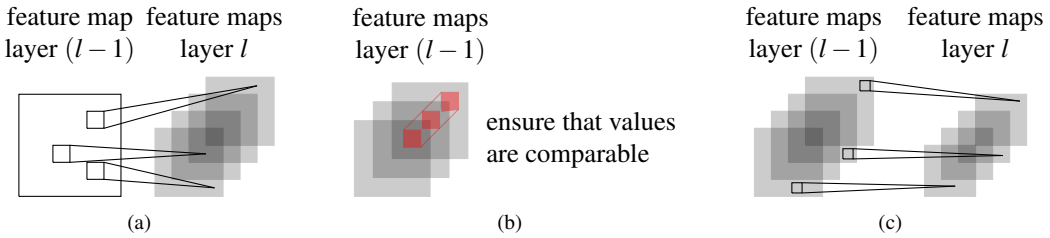

Figure 1: Illustration of a 1(a) convolutional layer; 1(b) local contrast normalization layer; and 1(c) pooling layer.

3.2.2 Non-Linearity and Rectification Layer

A non-linearity layer applies an activation function f component-wise on its input feature maps:

$$Y_i^{(l)} = f\left(Y_i^{(l-1)}\right), \quad \forall 1 \leq i \leq m_1^{(l)} = m_1^{(l-1)}. \tag{47}$$

Thus, the output of layer l is given by $m_1^{(l)} = m_1^{(l-1)}$ feature maps of size $m_2^{(l)} \times m_3^{(l)} = m_2^{(l-1)} \times m_3^{(l-1)}$. Common activation functions for convolutional neural networks are the logistic sigmoid, the hyperbolic tangent and the rectified linear unit $f(z) = \max\{0, z\}$. If $f = |\cdot|$, we refer to the layer as rectification layer which can also be interpreted as separate layer.

3.2.3 Local Contrast Normalization Layer

In [KSH12] also referred to as response normalization layer, a local contrast normalization layer aims to create competition among feature maps computed using different filters. Furthermore, contrast normalization layers can also be motivated using results from neuroscience [KSH12]. Krizhevsky et al. use brightness normalization:

$$\left(Y_i^{(l)}\right)_{r,s} = \frac{\left(Y_i^{(l-1)}\right)_{r,s}}{\left(\kappa + \lambda \sum_{j=1}^{m^{(l-1)}} \left(Y_j^{(l-1)}\right)_{r,s}^2\right)^{\mu}}, \quad \forall 1 \leq i \leq m_1^{(l)} \tag{48}$$

where κ, λ, μ are hyperparameters set using a validation set. Alternatively, the sum in Equation (48) may be restricted to those feature maps $Y_j^{(l-1)}$ adjacent to $Y_i^{(l-1)}$. The output of layer l is given by $m_1^{(l)} = m_1^{(l-1)}$ feature maps unchanged in size. The local contrast normalization layer is illustrated in Figure 1(b).

3.2.4 Pooling Layer

While LeCun et al. [LBD+89] use subsampling within convolutional layers (that is, the convolution in Equation (46) is not applied on every entry of the input feature maps), recent architectures prefer to use separate pooling layers to reduce the size of the feature maps and introduce invariance to noise and distortions. Similar to the aggregation step of local descriptors used by Ge et al. [GKS13], max pooling or average pooling are common:

Average pooling computes the average value within (non-overlapping) windows; while

Max pooling computes the maximum value within (non-overlapping) windows.

Pooling has been found to improve convergence and reduce overfitting [KSH12]. Note that pooling can also be used with overlapping filters, see [KSH12].

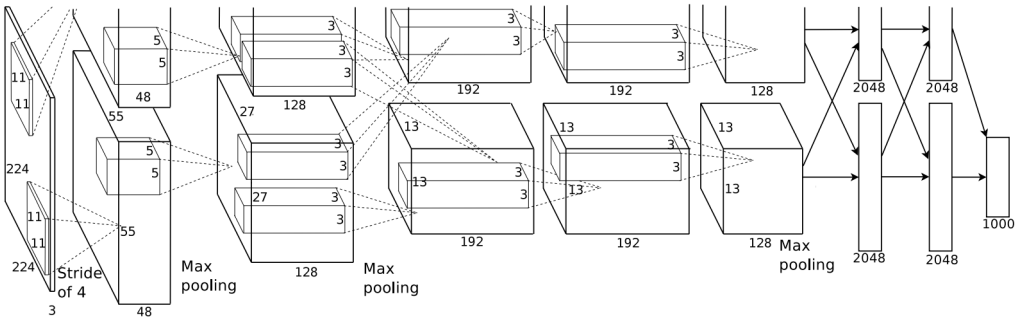

Figure 2: The architecture used by Krizhevsky et al. [KSH12]. The number $m_1^{(l)}$ of computed feature maps in layer l is indicated by the width of the corresponding cuboid, while the size $m_2^{(l)} \times m_3^{(l)}$ of the feature maps is indicated by the height and depth, respectively. In addition, the size of the filters $W_{i,j}^{(l)}$ is shown. For example, the input is given by a single gray-scale image of size 244×244. Using filters of size 11×11, the first convolutional layer computes $96 = 48 + 48$ feature maps of size 55×55. Furthermore, the filters in the first convolutional layer are applied in strides of 4 pixels. We note that the architecture has been parallelized and refer to [KSH12] for details.

3.2.5 Fully-Connected Layer

If layer l is a fully connected layer and layer $(l-1)$ one of the above layers, the input feature maps $Y_i^{(l-1)}$ are interpreted as $m_2^{(l-1)} \cdot m_3^{(l-1)}$-dimensional vectors and layer l computes:

$$y_i^{(l)} = f\left(z_i^{(l)}\right) = f\left(\sum_{j=1}^{m_1^{(l-1)}} \sum_{r=1}^{m_2^{(l-1)}} \sum_{s=1}^{m_3^{(l-1)}} w_{i,j,r,s}^{(l)} \left(Y_j^{(l-1)}\right)_{r,s}\right), \quad \forall 1 \leq i \leq m^{(l)}. \tag{49}$$

Note that, in contrast to convolutional layers, this layer already includes non-linearities. In the case that layer $(l-1)$ is a fully connected layer, Equation (49) can be applied analogously to Equation (43). In convolutional neural networks, both layer $(L-1)$ and layer L are usually fully connected layers and layer L uses the softmax activation function and is used for classification.

3.3 Architectures

Most architectures used in practice can be described using the layer types discussed above (for example [LBD+89, JKRL09]). For example, the architecture introduced by Krizhevsky et al. can be described as follows. Five convolutional layers with rectified linear unit non-linearity layers are followed by three fully connected layers [7]. Local contrast normalizaton layers are used after the first and second non-linearity layers (corresponding to the first and second convolutional layers). Overlapping max pooling layers are used after the first and second contrast normalization layer as well as after the fifth non-linearity layer. Although not formalized in Section 3.2.1, the convolutional layers are not necessarily fully connected to the previous layers, that is the sum in Equation (46) does not run over all input feature maps. This is due to computational reasons, see [KSH12]. The last layer uses softmax activation functions. Figure 2 shows the overall architecture including the specific fiilter and feature map sizes.

3.4 Available Implementations

The main problem of implementing and training deep convolutional neural networks is computational efficiency. Therefore, most implementations use one or more GPUs. For example, the original implementation

[7] The exact architecture can also be seen when consulting the reference model shipped with Caffe [JSD+14], see Section 3.4. Note that Caffe requires some extra layer types for training as well as pre-processing and includes dropout regularization, discussed in Section 3.5.1, as additional layer type.

by Krizhevsky et al. uses two Nvidida GTX 580 [8] – still, training on the ImageNet dataset [DDS+09] took several days. As a result, pre-trained models have become quite popular (that is, the weights of a trained network together with a specification of the network architecture are shared with the community). This is simplified by publicly available frameworks as for example Caffe [JSD+14], which comes with several pre-trained models using different architectures. Another popular convolutional neural network library is OverFeat [9].

3.5 Training

Training convolutional neural networks means to adjust the weights $\mathbf{W} = \{w_{i,j,r,s}^{(l)}\}$ according to a loss applied on layer L. Given a training set $\{(x_1,t_1),\ldots,(x_N,t_N)\}$ with $t_n \in \{1,\ldots,m^{(L)}\}$ being class labels, common loss functions for classification include the sum-of-squared error and the cross-entropy error. Instead, Krizhevsky et al. minimize the multinomial logistic loss:

$$E(\mathbf{W}) = -\frac{1}{m^{(L)}} \sum_{n=1}^{N} log\left(y_{t_n}^{(L)}\right) \tag{50}$$

where $y_{t_n}^{(L)}$ is expected to model the posterior probability $p(t_n|x_n)$ of sample x_n belonging to class t_n. These loss functions can be defined on the whole training set, as in Equation (50), on so-called mini-batches, that is on small subsets of the training set, or on individual training examples. Therefore, convolutional neural networks can also be trained online.

A common algorithm used for minimization is gradient descent, an iterative technique which, starting with an initial guess $\mathbf{W}[0]$, iteratively takes steps in the direction of the negative gradient (that is, the direction of the steepest descent in L_2 norm). Following [Bot12], and given a learning rate γ, gradient descent is implemented using the update step

$$\mathbf{W}[\tau+1] = \mathbf{W}[\tau] - \gamma \nabla E\left(\mathbf{W}[\tau]\right). \tag{51}$$

The chosen learning rate γ is crucial and usually has strong influence on the found minima. A widely used extension is gradient descent with momentum term which aims to avoid oscillation while using a high learning rate for fast training. The update step changes to:

$$\mathbf{W}[\tau+1] = \mathbf{W}[\tau] - \gamma \nabla E\left(\mathbf{W}[\tau]\right) + \nu(\mathbf{W}[\tau] - \mathbf{W}[\tau-1]) \tag{52}$$

where ν is the momentum parameter.

Stochastic gradient descent applies the update of Equation (51) on a subset of randomly selected samples. It can be shown that stochastic gradient descent converges under relatively loose conditions [Bot12]. In addition, stochastic gradient descent may be beneficial for large-scale learning [Bot12] and improve the generalization performance of the learned model [BB07].

For applying stochastic gradient descent, the gradient $\nabla E(\mathbf{W}[\tau])$ needs to be computed in each iteration. Therefore, Error Backpropagation has been proposed and allows to compute the gradient $\nabla E(\mathbf{W}[\tau])$ in $O(|\mathbf{W}|)$ [Bis95].

3.5.1 Regularization

Regularization is used to control the complexity of the model during training to prevent overfitting. For example, Krizhevsky et al. use weight decay and dropout as regularization during training. Weight decay is simple L_2-normalization of the weights and used because large weights result in poor generalization [Bis95]. The updated loss function can be written as:

$$\hat{E}(\mathbf{W}) = E(\mathbf{W}) + \lambda \|\mathbf{W}\|_2. \tag{53}$$

In contrast, dropout [KSH12] tries to decouple pairs of computation units (that is individual elements within the feature maps $Y_i^{(l)}$ or the computed vectors $y^{(l)}$). Each such unit is deactivated (that is, set to zero) with probability one half. The intention is to avoid subsequent layers or neighboring units being dependent on specific units.

[8] The implementation is available at `http://www.cs.toronto.edu/~kriz/`.

[9] Available at `http://cilvr.nyu.edu/doku.php?id=code:start`.

4 Neural Codes for Image Retrieval

As discussed in [Ben09], convolutional neural networks tend to learn useful high- and mid-level filters and the generated activations are rich representations of image content. Therefore, convolutional neural networks have been applied to many different tasks. Babenko et al. [BSCL14] use the generated feature activations as image representation for image retrieval. The resulting image representations are also termed neural codes.

4.1 Architecture, Training and Implementation

According to Babenko et al., the same architecture as presented in Section 3.3 was used together with an adapted version of the original implementation by Krizhevsky [KSH12] et al. Experiments were conducted using a model pre-trained on the ImageNet dataset [DDS+09] and a model re-trained on a custom dataset consisting of popular landmarks (referred to as the Landmark dataset). The dataset comprises $213,678$ images with 672 classes (corresponding to the different landmarks) and was collected using the Yandex search engine [10]. Although, the search queries have been published [11], the dataset is not directly reproducible. Furthermore, the implementation has not been made publicly available.
Training is done according to [KSH12], see Section 3.5, using stochastic gradient descent with a momentum parameter $\nu = 0.9$ and weight decay with $\lambda = 0.0005$. The learning rate is initialized with $\gamma = 0.01$ and manually divided by 10 whenever the error on a validation set stops decreasing. As Babenko et al. do not mention deviations from this procedure, we assume that the same approach was used for the re-trained model.

4.2 Compression

The feature activations in intermediate layers may be quite high-dimensional. Thus, Babenko et al. [BSCL14] experiment with two different approaches for dimensionality reduction: **PCA** and metric learning following the idea of [SPVZ13].

Large-Margin Dimensionality Reduction [SPVZ13]. Babenko et al. collect a dataset of $100k$ image pairs from the Landmark dataset using a simple image matching system (**SIFT** matching with Lowe's ratio test [Low04] and **RANSAC** validation). The goal is to learn a linear dimensionality reduction $P \in \mathbb{R}^{C' \times C}$ such that the distance of matching image pairs is below a learned threshold b while the distance of non-matching pairs is above this threshold. This can be formalized by the constraint

$$t_{n,n'} \left(b - d(Px_n, Px_{n'})^2 \right) > 1 \tag{54}$$

where $t_{n,n'} = 1$ if and only if the image pair $(x_n, x_{n'})$ is a matching pair. This can be accomplished by minimizing

$$E(P) = \sum_{n,n'}^{N} \max\{0, 1 - t_{n,n'} \left(b - (x_n - x_{n'})^T P^T P (x_n - x_{n'}) \right)\} \tag{55}$$

using stochastic gradient descent. In particular, given a sampled pair of image representations $(x_n, x_{n'})$ the dimensionality reduction $P[\tau]$ in iteration τ is updated only if

$$t_{n,n'} \left(b - (x_n - x_{n'})^T P[\tau]^T P[\tau] (x_n - x_{n'}) \right) > 1 \tag{56}$$

and according to

$$P[\tau + 1] = P[\tau] - \gamma t_{n,n'} P[\tau] (x_n - x_{n'}) (x_n - x_{n'})^T \tag{57}$$

where γ is the learning rate. Instead of an additional regularization term, Simonyan et al. use early stopping, that is minimization is stopped after a pre-defined number of iterations. Positive and negative pairs are sampled with equal probability.

[10] See `https://www.yandex.com/`.
[11] Available at `http://sites.skoltech.ru/compvision/projects/neuralcodes` (in Russian).

5 Experiments

In this section, we discuss several experiments presented by Babenko et al. [BSCL14]. However, first we introduce popular datasets and the general evaluation methodology employed in the image retrieval community.

5.1 Datasets

There are several commonly used datasets available within the image retrieval community. We briefly discuss two datasets used for evaluation of many approaches discussed in Section 2.1.1.

Oxford 5k (+100k, +1M) [PCI+07]. The Oxford dataset consists of $5,062$ images showing eleven different Oxford landmarks. The images were retrieved from Flickr [12] by querying directly for these landmarks and adding several distractor images using the query "Oxford" alone. For each landmark five queries are provided and for each query, four different lists are available: "Good", that is the queried object is visible in the image; "Ok", that is most of the object (more than 25%) is visible; "Junk", that is the object is heavily occluded (less than 25% visible); and "Absent", that is the object is not shown in the image. Furthermore, two set of distractor images are provided: $99,782$ images from the 145 most popular tags on Flickr (+100k); and $1,040,801$ images from the 450 most popular tags on Flickr (+1M). Duplicates with the original 5k datasets have been removed.

INRIA Holidays [JDS08]. The dataset comprises $1,491$ personal holiday images and comes with 500 distinct queries including ground truth results. Some of the images were taken explicitly to test the system against illumination changes and viewpoint changes. However, Babenko et al. manually bring all images in the correct orientation.

5.2 Mean Average Precision

Performance evaluation is usually based on a fixed number K of retrieved images per query. Given a query z_0 and retrieved images $Z = (z_1, \dots, z_K)$, we define:

True Positives TP: the number of images retrieved that are relevant to the query (that is, show the same object or scene);
False Positives FP: the number of images retrieved that are not relevant to the query;
False Negatives TN: the number of images which are relevant to the query but not retrieved.

Then, Precision and Recall are given as

$$\mathrm{Pre}(Z) = \frac{TP}{TP+FP} \quad \text{and} \quad \mathrm{Rec}(Z) = \frac{TP}{TP+FN}. \tag{58}$$

The Precision-Recall curve is a common instrument to visualize and understand the performance of a retrieval systems. Furthermore, this curve can be summarized in a single value: Average Precision which can be interpreted as the area under the curve. Let $\mathrm{Pre}_k(Z)$ and $\mathrm{Rec}_k(Z)$ be Precision and Recall up to the k-th retrieved image, respectively. Then, Average Precision is computed as [13]

$$\mathrm{AP}(Z) = \sum_{k=1}^{K} (\mathrm{Rec}_k(Z) - \mathrm{Rec}_{k-1}(Z)) \frac{(\mathrm{Pre}_{k-1}(Z) + \mathrm{Pre}_k(Z))}{2} \tag{59}$$

with $\mathrm{Rec}_0(Z) = 0$ and $\mathrm{Pre}_0(Z) = 1$. The Mean Average Precision is calculated to report the performance over a set of queries.

[12] See `https://www.flickr.com/`.
[13] See the provided source code at `http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/`.

	Oxford 5k	Holidays
[GRPV12]	–	0.774
[AZ13]	0.555	0.646
[GKS13]	–	0.767
[JZ14]	0.676	0.771
Pre-Trained on ImageNet [DDS+09]		
$y^{(15)}$	0.389	0.69
$y^{(16)}$	0.435	0.749
$y^{(17)}$	0.430	0.736
Re-Trained		
$y^{(15)}$	0.387	0.674
$y^{(16)}$	0.545	0.793
$y^{(17)}$	0.538	0.764

Table 1: Mean average precision for the Oxford 5k dataset and the Holidays dataset. Using the notation from Section 3, Babenko et al. use $y^{(17)}$, $y^{(16)}$ and $y^{(15)}$ as image representations. These layer correspond to the second and first fully connected layers, each with dimension $C = 4096$, and the last convolutional layer including non-linearity layer and max pooling layer, resulting in $C = 9216$. The results are compared to the following approaches: Gordo et al. [GRPV12] use Fisher Vectors on densily extracted **SIFT** descriptors; Arandjelović and Zisserman [AZ13] use **VLAD** with intra-normalization; Ge et al. [GKS13] use sparse-coding to combine **SIFT**, **DAISY** [TLF08] and Sparse-Coded Micro Features; and Jégou and Zisserman [JZ14] use Triangulation Embedding and Democratic Aggregation.

5.3 Results

As Babenko et al. do not provide the used implementation or the Landmark dataset, we were not able to reproduce their experimental results. Therefore, Table 1 summarizes results reported in [BSCL14]. In particular, they experiment with three distinct layers for image representation: $y^{(15)}$ corresponds to the last convolutional layer including the subsequent non-linearity layer and max pooling layer; $y^{(16)}$ and $y^{(17)}$ correspond to the first and second fully connected layer. On the Oxford 5k dataset, the pre-trained model shows poor performance and the re-trained model is not able to cope with the approach by Jégou and Zisserman [JZ14]. In contrast, both the pre-trained model and the re-trained model demonstrate state-of-the-art performance on the Holidays dataset. Further, we see that layer 16 performs best.
As discussed in Section 4.2, Babenko et al. experiment with **PCA** and Large-Margin Dimensionality Reduction and Table 2 shows the corresponding results. On both the Oxford 5k dataset and the Holidays dataset, the feature activations seem to be more robust to dimensionality reduction. Compared to the other approaches, also using **PCA** for dimensionality reduction, the compressed re-trained feature activations show significantly better performance. However, we also note that Large-Margin Dimensionality Reduction does not show a significant advantage over **PCA**. Lastly, figure 3 shows two retrieval examples.

5.4 Discussion

The experiments by Babenko et al. raise several questions. First of all, the choice of layer is only justified experimentally and the re-trained convolutional neural network has been trained towards the classification task, as well. Only in the conclusion, Babenko et al. consider training the network directly on image pairs. Second, Large-Margin Dimensionality Reduction has not been demonstrated on the re-trained neural network. As the approach does not demonstrate any significant increase in performance over **PCA** on the pre-trained model, this may imply that discriminative dimensionality reduction does not work well for convolutional neural networks. However, we note that Gordo et al. are able to significantly increase performance using Joint Subspace and Classifier Learning – a comparison not available in [BSCL14]. Finally, results for the compared approaches shown in Tables 1 and 2 are taken from the corresponding publications and Babenko et al. do not discuss the used evaluation pipeline in detail such that the reported results may not be comparable.

	Oxford 5k	Holidays
[GRPV12]	–	0.723
[GRPV12]*	–	0.764
[AZ13]	0.448	0.625
[GKS13]	–	0.727
[JZ14]	0.433	0.617
Pre-Trained on ImageNet [DDS+09]		
$y^{(16)}$ (**PCA**)	0.433	0.747
$y^{(16)}$ (Large-Margin)	0.439	–
Re-Trained		
$y^{(16)}$ (**PCA**)	0.557	0.789

Table 2: Mean average precision for the Oxford 5k dataset and the Holidays dataset using 128 dimensional image representations. Babenko et al. use layer $y^{(16)}$ (as it experimentally yields the best results) compressed to 128 dimensions using **PCA** and Large-Margin Dimensionality Reduction. The other approaches have been compressed as discussed in the corresponding publications: Gordo et al. [GRPV12] use **PCA** and Joint Subspace and Classifier Learning (marked with *); Arandjelović and Zisserman [AZ13] use **PCA**; Ge et al. [GKS13] use **PCA**; and Jégou and Zisserman [JZ14] use power-law normalization after **PCA** rotation and subsequently keep the first 128 dimensions (see [JZ14] for details).

Figure 3: Two retrieval examples (taken from [BSCL14]) on the Holidays dataset using the pre-trained model (on the top, respectively) and the re-trained model (on the bottom, respectively). In each row, the left-most image is the query image and correctly retrieved images are outlined in green. Babenko et al. point out that the lower example is a "rare exception" [BSCL14, p. 10] where the pre-trained model outperforms the re-trained model.

6 Conclusion

In the course of this report we presented current techniques in content-based image retrieval. In particular, we discussed local descriptors and their aggregation techniques in Section 2.1.1, as well as global descriptors in Section 2.1.2. Furthermore, after discussing convolutional neural networks in Section 3, we discussed the use of convolutional neural networks in image retrieval in Section 4. Finally, we presented experimental results following Babenko et al. [BSCL14] in Section 5.
Overall, we discussed many approaches related to the Bag of Visual Words model [SZ03] relying mostly on hand-crafted descriptors. Although some techniques make use of metric learning or similar (supervised or unsupervised) learning techniques, the embedding step and/or the aggregation step are heavily guided by manual processes, that is the whole retrieval system is explicitly engineered by hand. In strong contrast, Babenko et al. use convolutional neural networks to automatically learn most of these tasks. Although, the presented results are not fully convincing, especially on the Oxford 5k dataset [PCI$^+$07], this approach opens a new direction of research where several subproblems are solved fully by learning algorithms. Still, the image retrieval community will try to improve these results by adapting pre- or post-processing as well as network architecture.

6.1 Research Questions

We conclude that combining hand-crafted approaches with convolutional neural networks may be a promising future research direction. For example, different normalization techniques (for example [AZ13]) may improve performance. Furthermore, layer selection has only be done experimentally based on the mean average precision. In contrast, visualizing the energy of the image representations as done in [AZ13] may help to select suitable layers. In addition, the convolutional neural network was explicitly trained in a classification framework. Other architectures, as for example a Siamese architecture (see [BSCL14] for references), may be tailored directly towards the image retrieval task. Finally, techniques allowing to combine the information of multiple layers similar to [GKS13] may be interesting.

References

[AZ12] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition, Conference on*, pages 2911–2918, Providence, Rhode Island, June 2012.

[AZ13] Relja Arandjelović and Andrew Zisserman. All about VLAD. In *Computer Vision and Pattern Recognition, Conference on*, pages 1578–1585, Portland, Oregon, June 2013.

[BB07] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, pages 161–168, Vancouver, Canada, December 2007.

[Ben09] Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, November 2009.

[Bis95] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, New York, 1995.

[Bot12] Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 421–436. Springer, 2012.

[BSCL14] Artem Babenko, Anton Slesarev, Alexander Chigorin, and Victor S. Lempitsky. Neural codes for image retrieval. In *Computer Vision, European Conference on*, volume 8689 of *Lecture Notes in Computer Science*, pages 584–599, Zurich, Switzerland, September 2014. Springer.

[CPS+07] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Computer Vision, International Conference on*, pages 1–8, Rio de Janeiro, Brazil, October 2007.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, Conference on*, pages 248–255, Miami, Florida, June 2009.

[GKS13] Tiezheng Ge, Qifa Ke, and Jian Sun. Sparse-coded features for image retrieval. In *British Machine Vision Conference*, Bristol, United Kingdom, September 2013.

[GRPV12] Albert Gordo, José A. Rodríguez-Serrano, Florent Perronnin, and Ernest Valveny. Leveraging category-level labels for instance-level image retrieval. In *Computer Vision and Pattern Recognition, Conference on*, pages 3045–3052, Providence, Rhode Island, June 2012.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 1–6, Manchester, United Kingdom, September 1988.

[JDCSP10] Hervé Jégou, Matthijs Douze, C. Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition, Conference on*, pages 3304–3311, San Fransisco, California, June 2010.

[JDS08] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Computer Vision, European Conference on*, volume 5302 of *Lecture Notes in Computer Science*, pages 304–317, Marseille, France, October 2008. Springer.

[JDS11] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[JH99] Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, pages 487–493, Denver, Colorado, November 1999.

[JKRL09] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, International Conference on*, pages 2146–2153, Kyoto, Japan, September 2009.

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *Computing Research Repository*, abs/1408.5093, 2014.

[JZ14] Hervé Jégou and Andrew Zisserman. Triangulation embedding and democratic aggregation for image search. In *Computer Vision and Pattern Recognition, Conference on*, pages 3310–3317, Columbus, June 2014.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, Lake Tahoe, Nevada, December 2012.

[LBD+89] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, December 1989.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[MS04] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, October 2004.

[NS06] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, Conference on*, pages 2161–2168, New York, NY, June 2006.

[OT01] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, May 2001.

[PCI+07] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern, Conference on*, pages 1–8, Minneapolis, Minnesota, June 2007.

[PD07] Florent Perronnin and Christopher R. Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, Conference on*, pages 1–8, Minneapolis, Minnesota, June 2007.

[PISZ10] James Philbin, Michael Isard, Josef Sivic, and Andrew Zisserman. Descriptor learning for efficient retrieval. In *Computer Vision, European Conference on*, volume 6313 of *Lecture Notes in Computer Science*, pages 677–691, Heraklion, Greece, September 2010. Springer.

[Sin64] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Annals of Mathematical Statistics*, 35(2):876–879, June 1964.

[SPVZ13] Karen Simonyan, Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Fisher vector faces in the wild. In *British Machine Vision Conference*, Bristol, United Kingdom, September 2013.

[SZ03] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, International Conference on*, pages 1470–1477, Nice, France, October 2003.

[TLF08] Engin Tola, Vincent Lepetit, and Pascal Fua. A fast local descriptor for dense matching. In *Computer Vision and Pattern Recognition, Conference on*, pages 1–8, Anchorage, AK, June 2008.