Multi-View Product Image Search Using ConvNets Features

Muhammet Başţan · Özgür Yılmaz

Abstract Multi-view queries on a multi-view product image database with bag-of-visual words (BoWs) have been shown to improve the average precision significantly compared to traditional single view queries on single view databases. In this paper, we investigate the performance of deep convolutional neural networks (ConvNets) on multi-view product image search and compare to the classical BoWs. We used a simplified version of VGG network to train and extract global ConvNets image representations for retrieval. We performed experiments on the publicly available Multi-View Object Image Dataset (MVOD 5K) and concluded that (1) multi-view queries with ConvNets representations perform significantly better than single view queries, (2) ConvNets perform much better than BoWs and have room for further improvement.

1 Introduction

Traditionally image retrieval systems are based on single view images of objects in the database and in the query. However, there are some applications in which multi-view images of objects are available in the database. Product image search for online shopping is one such application. Figure 1 shows examples of typical multi-view images of some products from online shopping sites. An image search engine should leverage the availability of such multi-view object image databases to provide more accurate search results to the users.

With the ubiquity of smart phones with cameras, mobile product search is an emerging application area to provide users with an easier and richer shopping experience [8, 10, 11]. Users can easily take one or more photos of a product and search online shopping sites for visually similar products. This motivated the industry to develop mobile visual search applications, such as Google Goggles [13], CamFind [6], Amazon Flow [1] and Nokia Point & Find [18].

A mobile product image search system should consider the limitations of the mobile device and use the resources (CPU, memory, battery, network bandwidth) sparingly, while leveraging the user interaction potential to return more accurate results. It is crucial to rank the relevant results within top 10–20 results, because a mobile user will not have time to check a long result list. Multi-view queries can

E-mail: mubastan@gmail.com, yilmazozgur.kaan@gmail.com

Fig. 1 Multi-view images of some products from online shopping sites.

be helpful in such a context; as the user takes multiple photos of a product, the search system can process each image in the background and communicate with the server for further query processing on the database and return progressively better results using more query images.

Systems using "multi-image" queries try to improve retrieval accuracy using multiple images from the same category as the query image [2, 17, 25], the database contains single view images and multiple query images are not multi-view images of the query object. The first work to construct and use a multi-view object image database and multi-view queries is by Çalışır et al. [5]. They collected a multi-view product image dataset from online shopping sites, and showed through extensive experiments that multi-view queries on a multi-view database improves retrieval precision significantly compared to single view and multi-image queries. They used bag-of-visual-words (BoWs) as image representation and evaluated many early and late fusion approaches. The presented multi-view query model is independent of the image representations; it is not limited to BoWs. Using deep convolutional neural networks (ConvNets) representations as an alternative to BoWs was set as a future work.

ConvNets have proven to give state-of-the-art results in many computer vision problems, including image classification, retrieval [4, 14, 19, 20, 24] and multi-view recognition [23]. In this paper, we investigate the performance of ConvNets representations on multi-view product image search, using the multi-view query framework of [5] and compare the performance of the two representations (ConvNets and BoWs).

2 Multi-View Search with ConvNets Features

In this section, we describe the multi-view search using ConvNets features. Basically, we use the same multi-view query framework of [5] and use ConvNets

representations of images instead of BoWs. A convolutional neural network is first trained on labeled image datasets; the labels are object categories. Then, this ConvNet is run on the database and query images to extract features. Finally, these ConvNet features are used with Euclidean distance to rank the database objects with early or late fusion, as described below.

2.1 ConvNet Architecture

We use a simplified version of VGG networks [22]. The network architecture is shown in Figure 2. The input image size is $256 \times 256 \times 3$. The network has two convolutional layers with a kernel size of 3×3, followed by a max-pooling layer with kernel size 2×2 and stride 2; this is repeated five times. At the end of the convolutional layers, the image size is reduced to $8 \times 8 \times 128$. The convolutional layers are followed by three fully connected (FC) layers with 1024, 1024 and 45 neurons. Hence, there are 10 3×3 convolutional, 5 2×2 max pooling and 3 fully connected layers in the network.

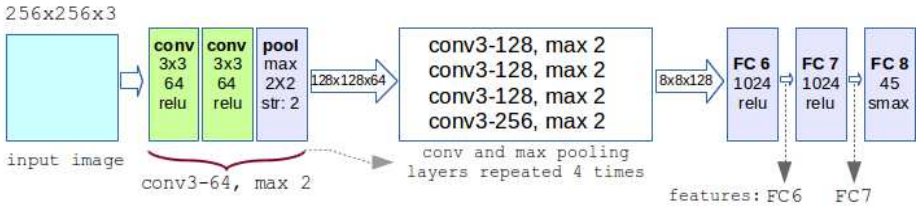

Fig. 2 Simplified VGG network architecture used in this work for global image feature extraction.

The features are extracted as the outputs of the fully connected layers FC6 and FC7. Following common practice, the extracted features are L_2 normalized to unit length (Figure 3) [3]; this improves the performance significantly.

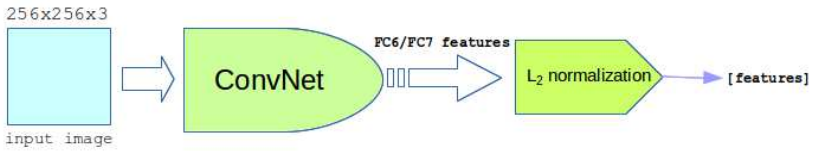

Fig. 3 ConvNets feature extraction and L_2 normalization.

2.2 Early Fusion

In early fusion, features extracted from multi-view images of an object are reduced to a single feature before applying the distance function (Figure 4). We used the

maximum and average functions for early fusion. If there are M multi-view images of an object, then the combined feature of an object is computed as

- *Maximum (EF-MAX):* $f_i = max(f_i^1, \ldots, f_i^M)$
- *Average (EF-AVG):* $f_i = \dfrac{\sum_{j=1}^{M} f_i^j}{M}$

where f_i is the i^{th} feature in feature vector f. Finally, the distance between the combined feature vectors are measured with Euclidean distance [3] to rank the database objects.

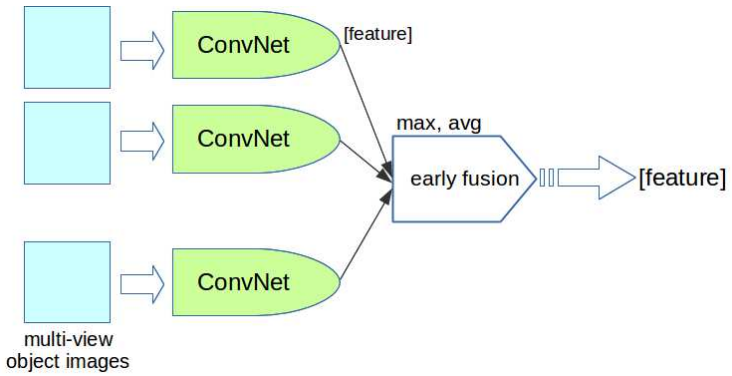

Fig. 4 Early fusion.

2.3 Late Fusion

In late fusion, first, the distances between all pairs of query and database images of an object are computed, then, the distances are combined into a single distance to rank the database objects (Figure 5). We use Euclidean distance to measure the dissimilarity between two image features. Once the distances are computed, there are several ways of late fusion: minimum, average, weighted average, etc.

The combined distance measure d between a query object, having M views, and database object, having N views, is computed in one of the following ways in our experiments, based on the results of [5] and called *image set distance/similarity*.

- *Minimum Distance (LF-MIN):* The distance between a query and database object is taken as the minimum of all $M \times N$ distances.

$$d = min(d_{ij})$$

- *Average Distance (LF-AVG):* The distance is computed as the average of all $M \times N$ distances.

$$d = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} d_{ij}}{M \times N}$$

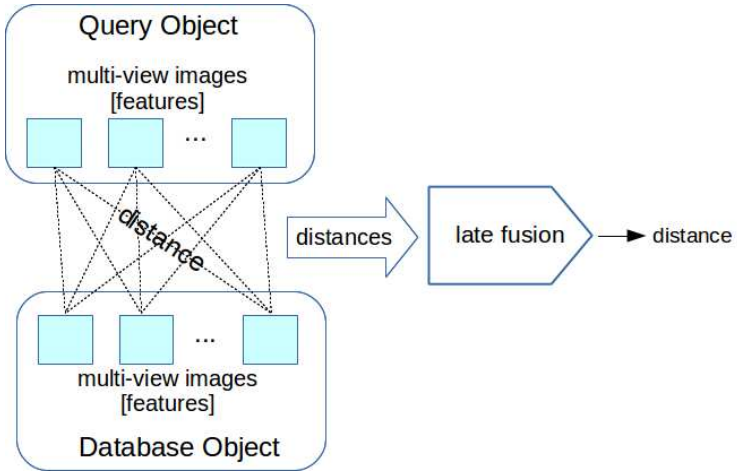

Fig. 5 Late fusion.

– *Weighted Average Distance (LF-WAVG, LF-IWAVG):* A weight w_{ij} is assigned to each distance d_{ij}. This weight can be proportional or inversely proportional $(1/d_{ij})$ to the distance.

$$w_{ij} = \frac{d_{ij}}{\sum_{i=1}^{M}\sum_{j=1}^{N} d_{ij}}$$
$$d = \sum_{i}^{M}\sum_{j}^{N} d_{ij} \times w_{ij}$$

– *Average of Minimum Distances (LF-MIN-AVG):* First, the minimum distance for each of M query images to N database object images is computed. Then, the average of M minimum distances is computed as the image set distance.

$$d = \frac{\sum_{i}^{M} \min(d_{i1}, \ldots, d_{iN})}{M}$$

– *Weighted Average of Minimum Distances (LF-MIN-WAVG):* This is the weighted average version of the previous method; the weighting is proportional.

$$d_i = \max(d_{i1}, \ldots, d_{iN})$$
$$w_i = \frac{d_i}{\sum_{i}^{M} d_i}$$
$$d = \sum_{i}^{M} w_i \times d_i$$

3 Dataset and Evaluation

We used the Multi-View Object Image Dataset (MVOD 5K) [5], publicly available at `www.cs.bilkent.edu.tr/~bilmdg/mvod/`. The database has 5K images from 45 different product categories (shoes, backpacks, eyeglasses, cameras, printers, guitars, coffee machines, etc.). Each object has at least two different images taken from different views (multi-view). The images mostly have a clean background and objects are positioned at the image center. Figure 6 shows sample images from the database.

Fig. 6 Sample images from the MVOD 5K database.

In addition to the 5K multi-view database images, the MVOD dataset has two sets of multi-view object queries:

Internet Queries. The multi-view query images are collected from online shopping sites, similar to the MVOD 5K database (Figure 7). There are 45 queries in the set, one query per object category. The images are similar to MVOD 5K, they mostly have clean background and objects are positioned at the image center.

Phone Queries. The multi-view query images are collected with a mobile phone in natural office, home or supermarket environments (Figure 8). The query set has 15 queries for 15 categories. The phone queries are more difficult than the Internet queries, since they have adverse effects, like background clutter and illumination problems.

We evaluated the retrieval performance in terms of average precision (AveP), as in [5, 21]. The average precision is calculated as shown below; k represents the rank in the result list and N is the length of the result list.

Fig. 7 Sample images from the MVOD Internet queries. All queries have multi-view images; here, only a single view image is shown for the sample queries.

Fig. 8 Sample images from the MVOD phone queries. All queries have multi-view images; here, only a single view image is shown for the sample queries.

$$P(k) = \frac{\text{relevant objects} \cap \text{first k objects}}{k}$$

$$rel(k) = \begin{cases} 1, & \text{if object k is relevant} \\ 0, & \text{otherwise} \end{cases}$$

$$AveP = \frac{\sum_{k=1}^{N}(P(k) \times rel(k))}{N}$$

Queries and query results are object-based; a list of objects are returned as the query result. A retrieved object is relevant (correct) if it belongs to the same object category as the query object [5, 21]. Although the evaluation is not based on instance retrieval, i.e., returning objects that are instances of the query object, it is desirable to rank visually similar database objects higher in the result list. There is no such publicly available multi-view object image dataset to use for instance-based retrieval evaluation.

The interpolated average precision graphs are obtained by averaging the average precisions over all queries.

4 Experiments

We performed extensive single and multi-view retrieval experiments on the MVOD 5K dataset and evaluated the performance of various fusion methods for multi-view queries. We also compared performance of ConvNets features to BoWs [5].

We implemented the ConvNets training and feature extraction using Google's TensorFlow library [12] and a higher level API TFLearn [9]. The categories in the dataset are mutually exclusive, only one object category is labeled in each image. We used one-hot encoding for the labels and minimized categorical cross entropy loss with Adam optimizer, which is available in TensorFlow. The ConvNet was first trained on Caltech 256 Object Categories dataset [15], with 20K images for 200 epochs, then refined on MVOD 5K database for 50 epochs. The images are resized to $256 \times 256 \times 3$, with padding for non-square images. Data augmentation (Gaussian blur with a maximum σ of 2.0, random left-right flip) is also employed.

After training, features are extracted from the database and query images using the fully connected layer outputs (FC6 and FC7) and L_2 normalized. Finally, the extracted global image features are used for retrieval with Euclidean distance. FC6 and FC7 features result in similar performance; the results below were obtained with FC7 features since they were slightly better. Feature extraction takes about 0.12 seconds per $256 \times 256 \times 3$ image, with TensorFlow on NVIDIA GEFORCE GT 650M GPU with 2GB memory, Intel CORE i7 2.4GHz CPU.

4.1 Results on the Internet Queries

Figure 9 shows interpolated average precision graphs for 45 Internet queries on the MVOD 5K database. Similar to the results of [5], multi-view queries with early and late fusion are significantly better than single view queries. Late fusion methods give similar results, with averaging methods being slightly better. Based on these results, early fusion approaches (maximum, average) can be preferred, since early fusion methods are faster than late fusion methods (with unoptimized, exhaustive search, single threaded Python implementation, early fusion maximum takes 0.132 seconds, early fusion average takes 0.148 seconds and late fusion average takes 0.319 seconds per query, on an Intel CORE i7 2.4GHz laptop with 32GB memory). Concatenating the FC6 and FC7 features did not result in meaningful improvement.

A network trained only on Caltech 256 [15] dataset for 200 epochs, without any refinement on MVOD 5K, performed rather poorly on the Internet queries (about half the performance). On the other hand, network trained only on MVOD 5K dataset performed similarly to the presented results. This may be because the Internet queries are very similar to the database.

4.2 Results on the Phone Queries

Figure 10 shows interpolated average precision graphs for 15 phone queries on the MVOD 5K database. As before, multi-view queries are significantly better than single view queries. Late fusion methods are better than early fusion, and averaging late fusion methods are slightly better than other late fusion methods.

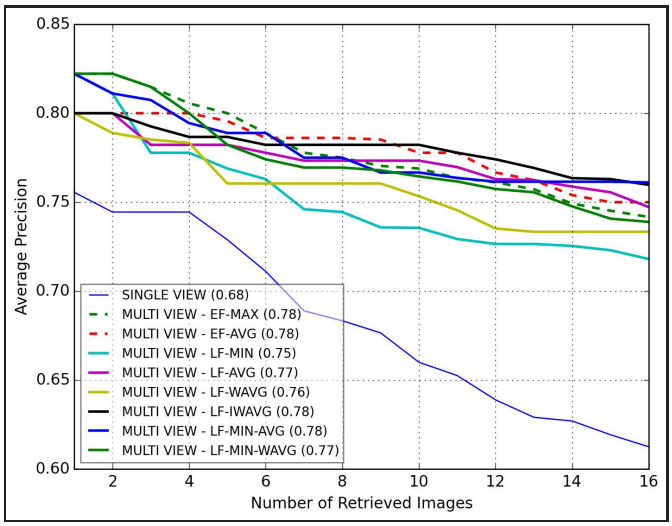

Fig. 9 Interpolated average precision graphs for 45 Internet queries on MVOD 5K. Numbers inside the parenthesis are the mean average precisions (mAP).

Concatenating the FC6 and FC7 features did not result in meaningful improvement.

A network trained only on Caltech 256 [15] dataset for 200 epochs, without any refinement on MVOD 5K, performed rather poorly on the phone queries as well (about half the performance). Moreover, a network trained only on MVOD 5K also performed poorly. This can be explained by the fact that, the database images have mostly clean background without any illumination problems, while the phone queries have cluttered backgrounds with adverse illumination effects. The network should be trained to account for such adverse effects.

4.3 ConvNets versus BoWs

In this section, we compare the retrieval performance of ConvNets features to the BoWs of [5], on both the Internet and phone queries. In [5], Harris and Hessian keypoint detectors with SIFT descriptors were used with a vocabulary size of $3K$ and hard assignment. The BoWs of Harris and Hessian were concatenated and a BoW histogram of size $6K$ was obtained; this is usually a very sparse histogram. Various similarity functions were evaluated to measure the similarity between BoW histograms and *Min-Max Ratio* was found to be the best. The BoW results given here (the best results reported in [5]) use the *Min-Max Ratio* function.

Figures 11 and 12 show the interpolated average precision graphs for ConvNets and BoWs on the Internet and phone queries, respectively. Both single view and multi-view average precision values of ConvNets are significantly higher on both query types. The improvement is especially huge on the Internet queries, almost doubling the average precision. Moreover, the dimensionality of ConvNets features (1024) is much lower than that of BoWs (6K). Figures 13–16 show sample Internet

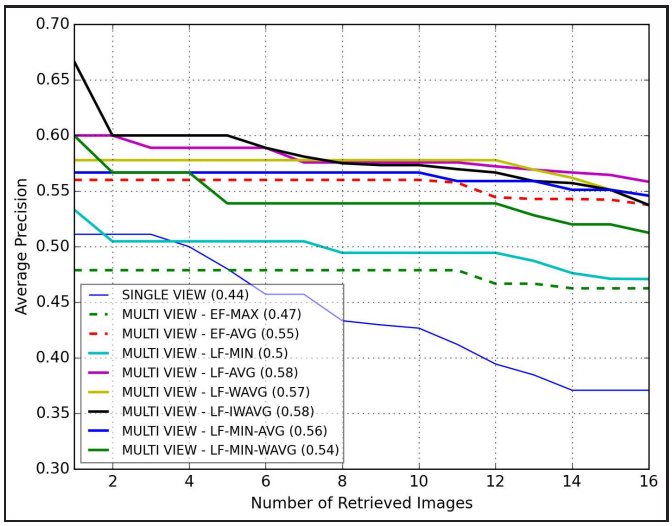

Fig. 10 Interpolated average precision graphs for 15 phone queries on MVOD 5K. Numbers inside the parenthesis are the mean average precisions (mAP).

and phone query results with BoWs and ConvNets. The samples are selected to be the same as those of [5] for comparison.

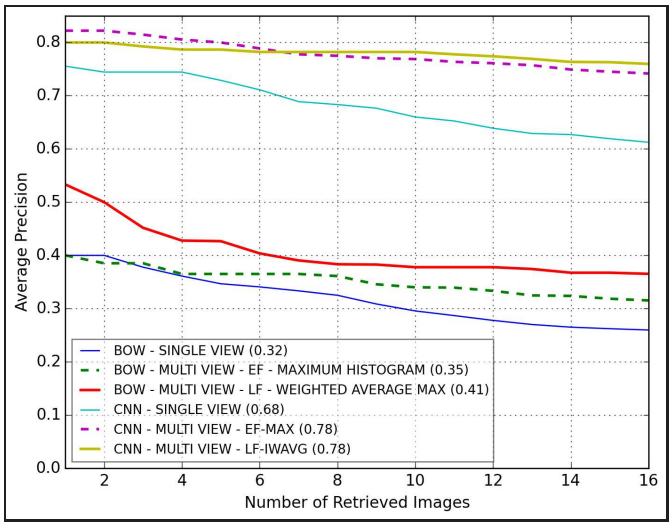

Fig. 11 Comparison of CNN and BoW features on 45 Internet queries, on MVOD 5K database. Numbers inside the parenthesis are the mean average precisions (mAP).

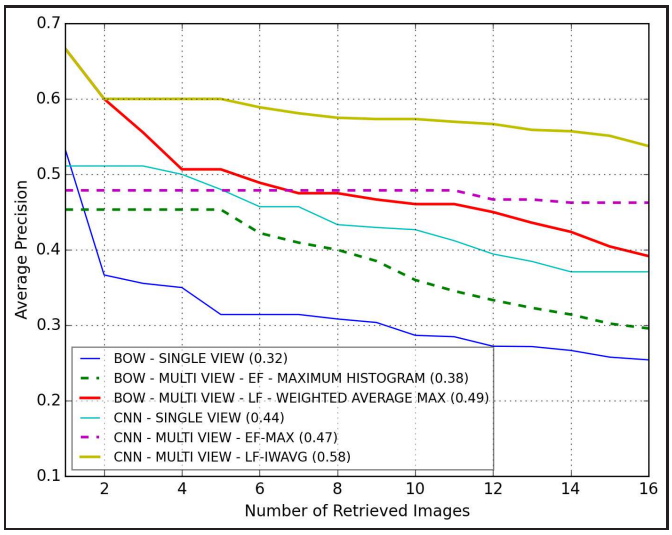

Fig. 12 Comparison of CNN and BoW features on 15 phone queries, on MVOD 5K database. Numbers inside the parenthesis are the mean average precisions (mAP).

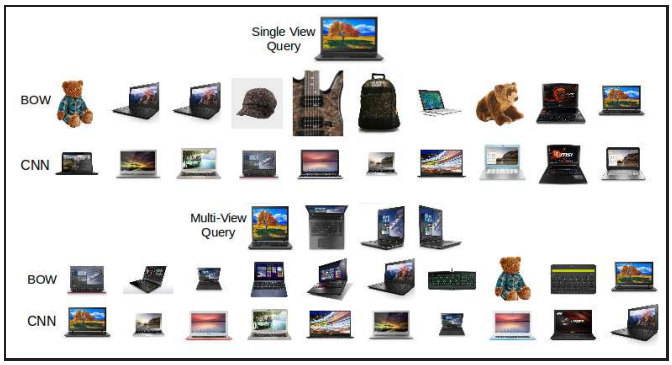

Fig. 13 Sample single and multi-view Internet query results with BoWs and ConvNets.

Fig. 14 Sample single and multi-view Internet query results with BoWs and ConvNets.

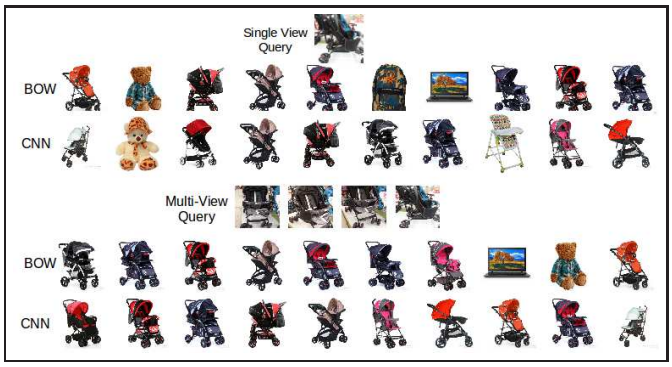

Fig. 15 Sample single and multi-view phone query results with BoWs and ConvNets.

Fig. 16 Sample single and multi-view phone query results with BoWs and ConvNets.

5 Conclusions

We investigated the performance of ConvNets features on multi-view product image search using various early and late fusion methods. Similar to the results of [5], multi-view queries on multi-view database result in significant performance improvement in terms of average precision. We also compared the ConvNets features with classical BoWs [5] and found ConvNets to be much better. Moreover, there is still room for performance improvement with ConvNets, by designing better networks and training on larger datasets. This work did not analyze the applicability of the designed network to mobile product search systems, which have memory and processing power limitations. It is left as a future work to design high performance ConvNets suitable for mobile product search systems; SqueezeNet-like networks [16, 7] with compression is a promising direction. Another direction is to build larger datasets for better training of ConvNets and more realistic performance evaluation.

6 Acknowledgements

This work was supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under 3501 with grant no 114E554.

References

1. A9.com, Inc.: Amazon Flow. http://www.a9.com/whatwedo/mobile-technology/flow-powered-by-amazon (2015). Accessed: 2015-07-27
2. Arandjelovic, R., Zisserman, A.: Multiple queries for large scale specific object retrieval. In: British Machine Vision Conference, pp. 92.1–92.11. BMVA Press (2012)
3. Azizpour, H., Sharif Razavian, A., Sullivan, J., Maki, A., Carlsson, S.: From generic to specific deep representations for visual recognition. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 36–45 (2015)
4. Babenko, A., Lempitsky, V.: Aggregating Deep Convolutional Features for Image Retrieval (2015)
5. Çalışır, F., Baştan, M., Ulusoy, Ö., Güdükbay, U.: Mobile Multi-View Object Image Search. Multimedia Tools and Applications (2016)
6. CamFind: CamFind. http://camfindapp.com (2015). Accessed: 2015-05-29
7. Cao, Y., Long, M., Wang, J., Zhu, H., Wen, Q.: Deep Quantization Network for Efficient Image Retrieval. In: AAAI Conference on Artificial Intelligence (2016)
8. Chen, D.M., Girod, B.: Memory-Efficient Image Databases for Mobile Visual Search. In: IEEE Multimedia, vol. 21, pp. 14–23 (2013)
9. Damien, A.: TFLearn. https://github.com/tflearn (2016). Accessed: 2016-08-08
10. Girod, B., Chandrasekhar, V., Chen, D.M., Cheung, N., Grzeszczuk, R., Reznik, Y.A., Takacs, G., Tsai, S.S., Vedantham, R.: Mobile Visual Search. IEEE Signal Processing Magazine **28**(4), 61–76 (2011)
11. Girod, B., Chandrasekhar, V., Grzeszczuk, R., Reznik, Y.A.: Mobile visual search: Architectures, technologies, and the emerging MPEG standard. IEEE MultiMedia **18**(3), 86–94 (2011)
12. Google: TensorFlow. https://github.com/tensorflow (2016). Accessed: 2016-08-08
13. Google, Inc.: Google Goggles. http://www.google.com/mobile/goggles (2015). Accessed: 2015-07-27
14. Gordo, A., Almazan, J., Revaud, J., Larlus, D.: Deep Image Retrieval: Learning Global Representations for Image Search. arXiv preprint arXiv:1604.01325 (2016)
15. Griffin, G., Holub, A., Perona, P.: Caltech-256 Object Category Dataset. Tech. Rep. CNS-TR-2007-001, California Institute of Technology (2007)
16. Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡1mb model size. arXiv preprint arXiv:1602.07360 (2016)
17. Lee, C.H., Lin, M.F.: A Multi-query Strategy for Content-based Image Retrieval. International Journal of Advanced Information Technologies **5**(2), 266–275 (2012)
18. Nokia: Point and Find. http://www.pointandfind.nokia.com (2015). Accessed: 2015-07-27
19. Paulin, M., Douze, M., Harchaoui, Z., Mairal, J., Perronin, F., Schmid, C.: Local Convolutional Features with Unsupervised Training for Image Retrieval. In: International Conference on Computer Vision (2015)
20. Paulin, M., Douze, M., Harchaoui, Z., Mairal, J., Perronin, F., Schmid, C.: Local Convolutional Features with Unsupervised Training for Image Retrieval. In: IEEE International Conference on Computer Vision, pp. 91–99 (2015)
21. Shen, X., Lin, Z., Brandt, J., Wu, Y.: Mobile Product Image Search by Automatic Query Object Extraction. In: European Conference on Computer Vision, pp. 114–127 (2012)
22. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR (2015)
23. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view Convolutional Neural Networks for 3D Shape Recognition. In: International Conference on Computer Vision, pp. 945–953 (2015)

24. Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., Wu, Y.: Learning fine-grained image similarity with deep ranking. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1386–1393 (2014)
25. Xue, Y., Qian, X., Zhang, B.: Mobile image retrieval using multi-photos as query. In: IEEE International Conference on Multimedia and Expo Workshops, pp. 1–4 (2013)