IPSJ Transactions on Computer Vision and Applications Vol.20 1–5 (Jan. 2028)

[DOI: 10.2197/ipsjtcva.20.1]

Express Paper

ILSVRC on a Smartphone

Yoshiyuki Kawano[1,a)] Keiji Yanai[1,b)]

Received: March 14, 2014, Accepted: April 24, 2014

Abstract: In this work, to the best of our knowledge, we propose a stand-alone large-scale image classification system running on an Android smartphone. The objective of this work is to prove that mobile large-scale image classification requires no communication to external servers. To do that, we propose a scalar-based compression method for weight vectors of linear classifiers. As an additional characteristic, the proposed method does not need to uncompress the compressed vectors for evaluation of the classifiers, which brings the saving of recognition time.
We have implemented a large-scale image classification system on an Android smartphone, which can perform 1000-class classification for a given image in 0.270 seconds. In the experiment, we show that compressing the weights to 1/8 leaded to only 0.80% performance loss for 1000-class classification with the ILSVRC2012 dataset. In addition, the experimental results indicate that weight vectors compressed in low bits, even in the binarized case (bit=1), are still valid for classification of high dimensional vectors.

Keywords: large-scale visual recognition, mobile image recognition

1. Introduction

In recent years, smartphones such as iPhone and Android phones have progressed greatly, especially in terms of computational power. A quad-core CPU is common as a smartphone's CPU, which is equivalent to a PC's CPU sold in a few year ago. Taking advantage of high computational power of a smart phone, a stand-alone real-time object recognition system on a smartphone has become possible [5], which can classify 100-class foods.

On the other hand, large-scale object recognition has drawn attention recently. ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) is a representative large-scale object recognition benchmark. In ILSVRC, the number of categories is 1000, and each category has more than 1000 training images.

Then, in this work, as a case study of large-scale object recognition on a mobile device, we propose a stand-alone mobile object recognition system which can recognize objects of 1000 categories of ILSVRC. To realize that, we have to overcome the memory limitation in a smartphone environment which is different from a PC where a large amount of memory such as 128GB is available. To implement 1000-class visual classification on a smartphone, we propose an effective compression method on the weight vectors of linear classifiers to fit the trained parameters to the memory limitation of a smartphone.

As a recognition method for ILSVRC, Deep Convolutional Neural Network (DCNN) [6] became more popular since 2012. However, DCNN for large-scale object recognition requires large-size network and a huge number of parameters. According to [6], the DCNN which won ILSVRC2012 have 60 million parameters. The other common method than DCNN for large-scale object recognition is a combination of Fisher Vector (FV) [9] and linear classifiers. In fact, a FV-based method requires much more memory than DCNN. When the codebook size is 256 and the level of the spatial pyramid [8] is 1, totally the dimension of feature vectors is 163,840. Since the weights of trained linear classifiers for 1000 classes are needed to be stored, the total number of the parameters is 163 million. Currently, it is unrealistic to implement FV and linear classifiers as well as DCNN on a mobile device as they are, because of the memory limitation.

In fact, in case of an Android smartphone, the size of an application file (APK file) including both execution binary code and data is limited to 50MB at most. Practically, we can assign only about 40MB for data area, because we have to keep a space for run-time binary code. Therefore, in this paper, we assume the size limitation is 40MB. Note that in our work, recognition on 1000 categories is carried out on a smartphone, while training on 1000 categories is performed on PCs. Therefore, an application file has to contain all the trained parameters of classifiers and the other parameters related to feature coding such as Gaussian Mixture Models (GMMs) for Fisher Vector (FV) coding.

To implement 1000-class visual classification on a smartphone, we propose to compress the trained weight vectors of linear classifiers to fit them to the memory limitation. Linear classifiers in one-vs-rest manner are an economical method for multi-class classification, which require weight vectors the number and the dimension of which equals the number of classes to be classified and the dimension of feature vectors, respectively. However, we needs a large amount of memory to store them, when we use high-dimensional Fisher Vectors for higher classification performance. Of course, we can reduce the dimension of FV. However, it will bring performance reduction [10]. To keep high classifica-

[1] Department of Informatics, The University of Electro-Communications, Tokyo,
1-5-1 Chofugaoka, Chofu-shi, Tokyo 182–8585 Japan
[a)] kawano-y@mm.inf.uec.ac.jp
[b)] yanai@inf.uec.ac.jp

tion performance, dimension reduction should be avoided. Then, in this paper, we compress each element of the weight vectors without compressing their dimension.

In image classification, compression of classifier weights for classification has never discussed before, while compression of feature vectors for training time has addressed by Sanchez et al. [10]. They compressed feature vectors coded by Fisher Vector [9] with Product Quantization (PQ) [3] in training time to store a large number of feature vectors on memory at once. They never compressed FV as well as weight vectors in classification time, because they performed classification on a PC where enough memory is available. On the other hand, in our work, we carry out classification on a smartphone where memory is limited. Note that we perform training of linear classifiers using PQ in the same as [10]. We use AROW [2] for an online learning method of linear classifiers, while they used Stochastic Gradient Descent (SGD).

In the experiments, at first, we compare image features to select suitable combination of image features referring to [5]. Next, we analyze classification performance versus compression rates of weight vectors of linear classifiers with the ILSVRC2012 dataset and a 100 food image dataset.

The contributions in this paper are follows:

(1) We propose a scalar-based compression method for weight vectors of linear classifiers, which requires no uncompressing of the compressed vectors for evaluation of the classifiers.
(2) Compressing the weights to 1/8 leads to only 0.80% performance loss for 1000-class classification with the ILSVRC2012 dataset.
(3) We have proved that stand-alone large-scale image classification is possible on a consumer smartphone without communication to external servers.

2. Related Work

As commercial services on image recognition for smartphones, Google Goggles[*1] is widely well-known. Since it is based on specific object recognition methods, recognition of generic objects is impossible. Kumar *et al.*[7] proposed a mobile system which recognizes 184 plant species using curvature based shape feature peculiar leaf. In the system, leaf recognition was performed on a server. Su et al. [11] also proposed a server-based image classification system with a 137-class dataset which was a subset of the ILSVRC2012 dataset. They reduced amount of data sent to a recognition server with client-side feature extraction and sending a thumbnail image instead of an original one to the server. These three systems basically employs server-side image recognition, while our goal is to show that no recognition sever is needed for mobile image classification even for 1000 classes.

A few mobile systems without recognition servers have been proposed so far. Among them, the 100-class food image classification system proposed by Kawano et al. [5] is the state-of-the-art to the best of our knowledge, which adopted one-vs-rest liner SVM and Fisher Vector (FV) with HOG patch and Color patch features. The system can classify one food image only in 0.065 seconds. However, due to the memory limitation, the dimension of FV in the system was about 1500, which is very small compared to the state-of-the-art systems on PCs [10]. Especially, in large-scale image classification, small dimension features will cause a large loss in classification performance. Then, in this work, we compress each element of the weight vectors without reducing the dimension of feature vectors.

3. Proposed Method

In this section, first, we describe feature vectors, and next explain a classification method including how to compress weight vectors which is our major contribution in this work.

3.1 Feature Vectors

We use RootHOG patches, Color patches and uniform LBP as local features, and code them into Fisher Vectors (FV) [9] after applying PCA.

3.1.1 Histogram of Oriented Gradient (HOG) Patch

We use HOG as a local descriptor instead of SIFT and SURF, since it is simple and very fast to extract. HOG patches can be densely-sampled effectively by preparing a table storing a gradient orientation and its degree on each pixel in advance.

We extract HOG features in dense sampling. The HOG we used consists of 2×2 blocks (totally four blocks). We extract gradient histogram regarding eight orientations from each block. The total dimension of a HOG Patch feature is 32. After extraction of HOG patches, we convert each of them into a "RootHOG". It is an element-wise square root of the L1 normalized HOG, which is inspired by "RootSIFT" [1]. In fact, in the preliminary experiments, RootHOG leaded to the better performance than original HOG.

3.1.2 Color Patch

We use mean and variance of RGB value of pixels as a Color patch feature. We divide a local patch into 2×2 blocks, and extract mean and variance of RGB value of each pixel within each block. Totally, we extract 24-dim Color Patch features.

3.1.3 Local Binary Pattern (LBP) Patch

LBP represents texture patterns. LBP can be extract effectively by preparing a pixel-wise LBP table over whole an image. We used 59-dim uniform LBP.

3.2 Fisher Vector

Fisher Vector [9] is the state-of-the-art coding method which utilizes a high order statistic of local feature vectors to decrease quantization error other than Deep Convolution Neural Network in ILSVRC. In general, the dimension of FV for large-scale classification becomes very high (sometimes over million) to achieve higher performance.

According to [9], we encode local descriptors into Fisher Vector. We choose a probability density function (pdf) as a Gaussian mixture model (GMM) with diagonal covariances. Regarding Fisher Vector, we use only term on mean according to [4]. The gradient with respect to the mean is defined as follows:

[*1] http://www.google.com/mobile/goggles/

$$\mathcal{G}^X_{\mu,i} = \frac{1}{T\sqrt{\pi_i}} \sum_{t=1}^{T} \gamma_t(i) \left(\frac{x_t - \mu_i}{\sigma_i} \right), \tag{1}$$

where x_t and T are a feature vector of a local descriptor and the number of local descriptors, respectively. $\gamma_t(i)$ represents the probability of x_t belonging to the i-th component:

$$\gamma_t(i) = \frac{\pi_i \mathcal{N}(x_t|\mu_i, \Sigma_i)}{\sum_{j=1}^{N} \pi_j \mathcal{N}(x_t|\mu_j, \Sigma_j)} \tag{2}$$

Finally, gradient $\mathcal{G}^X_{\mu,i}$ are calculated for all the Gaussian. Fisher Vector is obtained as their concatenation.

To improve recognition accuracy, we apply power normalization ($\alpha = 0.5$) and L2 normalization [9]. In addition, Spatial Pyramid [8] with level 1 (1x1+2x2) is used for FV coding.

3.3 Linear Classifier

In large-scale image classification, in general, batch learning of classifiers is impossible, because it requires too much memory to store training samples and too much time to optimize training parameters. Instead, online learning is commonly used, since it updates training parameters one by one for each of the training samples and requires only one sample at a time. In ILSVRC2012, most of the team used SGD or Passive Aggressive (PA) as an online learning method, while we use AROW [2].

3.3.1 Adaptive Regularization of Weights (AROW)

AROW [2] is an online learning method of linear classifiers, which is robust to noise label. This property is suitable for a large-scale data, especially data gathered from the Web. We use AROW++[*2] with slight modification as an implementation of AROW.

AROW has a weight vector μ and a confidence matrix Σ as training parameters. μ is used for classification by dot-product with a feature vector, and the eigenvalues of Σ is monotonically decreasing as updating.

In each iteration, we pick one sample $\mathbf{x}_t$, and compute margin $m_t = \mu_{t-1} \cdot \mathbf{x}_t$ and confidence $v_t = \mathbf{x}_t^{\mathrm{T}} \Sigma_{t-1} \mathbf{x}_t$. If $m_t y_t$ is negative, update μ and Σ by the following equations:

$$\mu_t = \mu_{t-1} + \alpha_t \Sigma_{t-1} y_t \mathbf{x}_t \tag{3}$$
$$\Sigma_t = \Sigma_{t-1} - \beta_t \Sigma_{t-1} \mathbf{x}_t \mathbf{x}_t^{\mathrm{T}} \Sigma_{t-1}, \tag{4}$$

where $\beta_t = 1/\mathbf{x}_t \Sigma_{t-1} \mathbf{x}_t + r$ and $\alpha_t = max(0, 1 - y_t \mathbf{x}_t^{\mathrm{T}} \mu_{t-1})\beta_t$.

In training time, training samples are re-ordered randomly, and we iterate training of the parameters ten times to avoid bad effect by sample order. In case of 1000-class classification, we use all the samples in the target class as positive samples, while we sample 100 images from each of the other 999 classes, and use totally 999,000 images as negative samples. For one-vs-rest classification for 1000 classes, we train weight vectors of AROW for each of 1000 classes independently using a PC cluster.

In training time, we have to store one million samples coded by FV on memory in spite of using online learning, because practically training is carried out over all the samples iteratively to prevent dependency on the order of training samples. Their amount is too large even for a PC. Then, we compress training samples with Product Quantization (PQ) [3] following to [10]. Only the sample is uncompressed, when it is needed to update the parameters.

Sanchez et al. [10] showed that compression of feature vectors by 1/32 leaded almost no performance loss in case of vector-quantizing every 8 dimensions of feature vectors coded by FV with 2^8 centroids. Therefore, we adopt Product Quantization (PQ) [3] in training time.

[*2] https://code.google.com/p/arowpp/

3.4 Compression of Weight Vectors for Classification on a Smartphone

As mentioned in Section 1, 1000-class classification with one-vs-rest linear classifiers and high dimensional Fisher Vector requires a large amount of classifier weights. It it the problem to be solve for implementing 1000-class visual classification on a smartphone.

In this paper, we propose a scalar-based compression method for weight vectors of linear classifiers. This method has a good characteristic that we do not need to uncompress the compressed vector when evaluating the corresponding classifier.

Empirically, most of the element values of the weight vectors of linear classifiers are distributed between -1 and 1. Then, we restrict the range of an element value w_i within $(1, -1]$ by the following rules:

$$w'_i = \begin{cases} 0.999999 & (w_i \geq 1) & \text{(5a)} \\ w_i & (-1 < w_i < 1) & \text{(5b)} \\ -1 & (w_i \leq -1) & \text{(5c)} \end{cases}$$

Next, we compress it into n-bit representation.

$$w''_i = \lfloor w'_i \times 2^{n-1} + 2^{n-1} \rfloor, \tag{6}$$

Note that $\lfloor x \rfloor$ represents a maximum integer value which is not more than x. With this conversion, w''_i is represented as an integer value within $[0, 2^n - 1]$, which can be expressed in n bits. Note that there is room for improvement, because this conversion is relatively straight-forward. However, this has worked very well in the experiments.

To classify images in one-vs-rest manner, only relative descending order of the output values of the classifiers for the same Fisher Vector is important. Therefore, we can omit to reconstruct original values, it is enough for classification to compute a dot product between a compressed weight vector and a FV-coded feature vector. Even reconstruction of the sign of the unsigned compressed values is not needed, because the constant values added to make the values unsigned can be ignored for comparison of the output values in one-vs-rest classification. This enables us to save recognition time on a smartphone.

3.5 Implementation on a Smartphone

We implement a system as a multi-threaded system assuming a 4-core CPU smartphone is a target. Refer to Kawano et al. [5], we parallelize feature extraction for RootHOG Patch and Color Patch features. Extraction of local descriptors, applying of PCA with them, encoding them into Fisher Vector with power normalization and L2 normalization, and evaluation of 1000 classifiers are carried out over 2 cores in parallel for each feature, totally over

4 cores in parallel. Moreover, in the same as [5], we compute all the constant values which can be calculated in advance. Regarding the weights of the classifiers, we store two 4-bit compressed vectors into one-byte memory, and separate them on demand in classification time.

4. Experiments

4.1 Experimental Setup

In the experiment, as a large-scale image dataset, we use the ILSVRC2012 dataset [*3] which consists of 1000 classes. Since our objective is classifying 1000 classes on a smartphone in a practical speed (less than 1 second), we set parameters by regarding recognition speed as more important than performance. As an additional experiment, we use 100-class food dataset, UEC-Food100 [*4].

We evaluate the results of 1000-class classification with the top-5 error rate, which represents the rate that no ground-truth label is found in the top-5 classes in the descending order of the output values of the classifiers. For UEC-Food 100, we evaluated the results with the top-k classification rate varying k from 1 to 10. All the evaluations except for execution time were carried out on a PC.

Regarding image features, we prepare three kinds of features, Color-patch, RootHOG-patch and LBP, each of which represents color, gradient, and texture pattern, respectively. We sampled them densely in every 5 pixels with two scales. Before coded by Fisher Vector, they are applied with PCA and converted into 32-dim vectors in case of RootHOG-patch and LBP, and 24-dim vectors in case of Color-patch. Although actually the dimension of feature vectors for RootHOG-patch and Color-patch are not reduced by applying PCA, PCA is still important for whitening of feature vectors before FV coding. For FV coding, we use the GMM with 64 Gaussians and Spatial Pyramid level 1 (1x1+2x2). As results, the total dimensions of FV are 10240 for RootHOG-FV and LBP-FV and 7680 for Color-FV.

In training time, we train the classifier weights with AROW for 1000-class using a PC cluster, and compress them into n-bit representation. In classification time, we calculate dot-product of the compressed vector and the FV-coded feature vector for all of 1000 classes, and select the top-5 classes in terms of descending order of the evaluation value of the classifiers. In case of using two features, we simply add two output values on each class without any weighting.

4.2 1000-class classification

At first, we compare three kinds of features, Color-FV, RootHOG-FV and LBP-FV. Combining all the three features is impractical for implementing on a smartphone in terms of running time as well as memory. We assume extraction of two kinds of features are carried out in parallel by assigning each of them to two CPU cores. Note that we did not use weight compression in this experiment.

Fig.1 shows top-5 error rates for 1000-class classification of the ILSVRC2010 dataset. Among the single features, RootHOG-FV

[*3] http://www.image-net.org/challenges/LSVRC/2012/
[*4] http://www.foodcam.mobi/dataset/

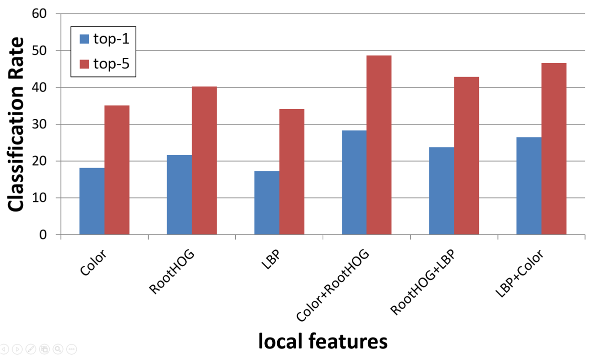

Fig. 1 Top-1 and Top-5 classification rates for the ILSVRC2012 dataset with three kinds of features and their combinations.

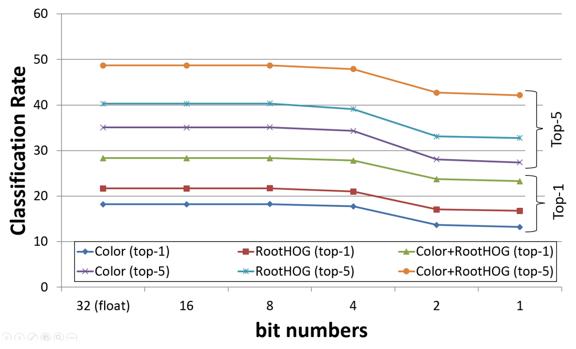

Fig. 2 Top-1 and Top-5 classification rates for the ILSVRC2012 dataset by the proposed method with n-bit compressed weight vectors (n=1, 2, 4, 8, 16). “32 (float)“ means the result with no weight compression.

achieved the best result, 59.7%. Among the combination of two features, Color+RootHOG was the best, 51.3%, and LBP+Color was the second best, 53.4%.

Next, we compare the results when varying the bits for weight compression from 1 to 32. In the case that the number of bit is 32, we use the weight vectors presented in “float”, which means no compression. Fig.2 shows the top-5 error rates by combination of Color-FV and RootHOG-FV and their single features. From 32 bits to 4 bits, no prominent performance drops were observed. The error rate was increased by only 0.8 points in case of 4-bit compression with combined features compared to original floating weights (32bit). On the other hand, in case of 2 and 1 bits, relatively larger performance drops by more than 7 points were observed. However, even in case of 1 bit which corresponds to binarization of weight vectors, the result is still meaningful. This is the surprising result which we have never expected.

Although the top-5 error rates 52.1% is less than the lowest results at ILSVRC2012, it corresponds to the fifth rank among eleven participants at ILSVRC2010.

4.3 Recognition Time

We use Samsung Galaxy NoteII (1.6GHz 4-core CPU, 4 threads, Android 4.1) for measuring recognition time for 1000 classes. In case of 4-bit compression, it took 0.270 seconds in one-time recognition of 1000 classes of ILSVRC. The proposed system achieved a reasonable recognition time as an image recognition engine of an interactive mobile application.

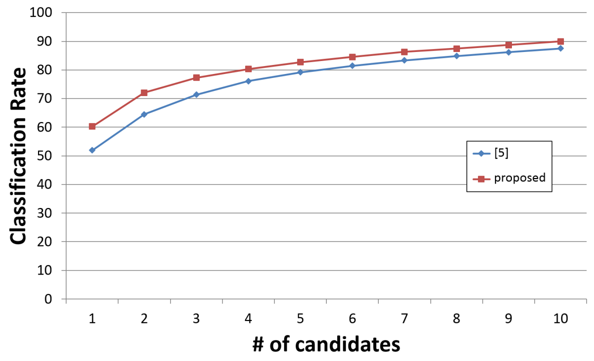

Fig. 3 Top-*k* classification rates for the UEC-Food100 dataset by the proposed method and Kawano et al. [5] (k=1,..,10).

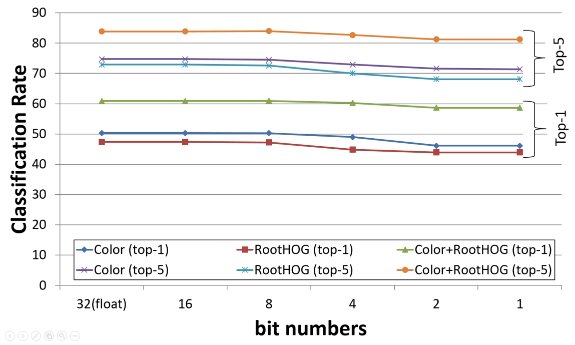

Fig. 4 Top-1 and Top-5 classification rates for the UEC-Food100 dataset by the proposed method with *n*-bit compressed weight vectors (n=1, 2, 4, 8, 16, 32(float)).

4.4 Required Memory

The classifier weight vectors occupy most of the memory area, which consists of 1000 7680-dim weights for Color-FV and 1000 10240-dim weights for RootHOG-FV. They totally needs 71.7MB to store without compression. By compressing them in 4-bit representation, their size is reduced to one eighth, about 9.0MB. Since 50MB is the limit size of the Android application, 9.0MB is enough small, while 71.7MB is too large to fit it.

4.5 Other dataset

As another dataset which is not so large as ILSVRC, we use UEC-FOOD100 which consists of 100-class food images. We compare classification rate of multi-class classification with [5] which achieved the best results on the dataset. The setting for feature extraction is the same for ILSVRC2012, except for the size of GMM for FV coding and additional usage of Spatial Pyramid [8].

In [5], the size of GMM was 32, and as feature vectors they used 2048-dim HOG-FV and 1536-dim Color-FV due to memory limitation. On the other hand, in this work, we use 32768-dim RootHOG-FV and 24576-dim Color-FV, the dimensions of which were raised 16 times by using the doubled size of GMM with 64 Gaussians and Spatial Pyramid with 1x1+2x2+1x3. This can be possible by introducing weight vector compression.

Fig.3 shows the top-*k* classification rates by our method with 4-bit compression and [5]. The proposed method achieved the 60.25% top-1 classification rate, which outperformed [5] by 8.4 points.

Fig.4 shows the result by our method with the different number of bit representation of the weight vectors. In case of 4-bit, the rate was dropped by only 1.2% compared to the case with no compression (32-bit float).

This result indicates that a technique to compress weight vectors of linear classifiers is effective not only for large-scale classification but also for even small-scale classification on an environment with memory limitation such as a smartphone.

5. Conclusions and Future Works

In this work, to the best of our knowledge, we proposed the first stand-alone large-scale image classification system running on an Android smartphone [*5].

We have proved that mobile large-scale image classification requires no communication to external servers. To realize that, we proposed a scalar-based compression method for weight vectors of linear classifiers. As an additional characteristics, the proposed method does not need to uncompress the compressed vectors for evaluation of the classifiers, which brings the saving of recognition time. In the experiment, we showed that compressing the weights to 1/8 leaded to only 0.80% performance loss for 1000-class classification with the ILSVRC2012 dataset. In addition, the experimental results indicated that weight vectors compressed in low bits, even in the binarized case (bit=1), are still valid for classification of high dimensional vectors.

For future work, we will try to implement a 10k-class classification system. To do that, we plan to split and share the weight vectors to achieve additional reduction of required memory. Reducing memory is helpful to increase the dimension of feature vector as well, which will bring performance improvement. Therefore, we will compress weight vectors more and more for better performance.

References

[1] Arandjelovic, R. and Zisserman, A.: Three things everyone should know to improve object retrieval, *CVPR*, pp. 2911–2918 (2012).
[2] Crammer, K., Kulesza, A. and Dredze, M.: Adaptive regularization of weight vectors, *Machine Learning*, Vol. 91, No. 2, pp. 155–187 (2013).
[3] Jégou, H., Douze, M. and Schmid, C.: Product quantization for nearest neighbor search, *IEEE Trans. on PAMI*, Vol. 33, No. 1, pp. 117–128 (2011).
[4] Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P. and Schmid, C.: Aggregating local image descriptors into compact codes, *IEEE Trans. on PAMI*, Vol. 34, No. 9, pp. 1704–1716 (2012).
[5] Kawano, Y. and Yanai, K.: Rapid Mobile Object Recognition Using Fisher Vector, *Proc. of Asian Conference on Pattern Recognition* (2013).
[6] Krizhevsky, A., Sutskever, I. and Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks., *NIPS* (2012).
[7] Kumar, N., Belhumeur, P., Biswas, A., Jacobs, D., Kress, W., Lopez, I. and Soares, J.: Leafsnap: A Computer Vision System for Automatic Plant Species Identification, *ECCV* (2012).
[8] Lazebnik, S., Schmid, C. and Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, *CVPR*, pp. 2169–2178 (2006).
[9] Perronnin, F., Sánchez, J. and Mensink, T.: Improving the Fisher Kernel for Large-Scale Image Classification, *ECCV* (2010).
[10] Sánchez, J. and Perronnin, F.: High-dimensional signature compression for large-scale image classification, *CVPR*, pp. 1665–1672 (2011).
[11] Su, Y. C., Chiu, T. H., Chen, Y. Y., Yeh, C. Y. and Hsu, W. H.: Enabling low bitrate mobile visual recognition: a performance versus bandwidth evaluation, *ACM MM*, pp. 73–82 (2013).

[*5] For iOS, DNN-based 1000 recognition app., JetPac Spotter, has been released this April. However, no Andriod app. has been released.