Low-Bit Representation of Linear Classifier Weights for Mobile Large-Scale Image Classification

Yoshiyuki Kawano Keiji Yanai
Department of Informatics, The University of Electro-Communications, Tokyo, Japan
{kawano-y,yanai}@mm.inf.uec.ac.jp

Abstract

In this paper, we propose an effective method to implement a system of large-scale visual recognition where the number of classes is more than 1000 on mobile devices. Because the size of memory and storage on mobile devices such as smartphones is limited, the size of image recognition application should be as small as possible. To save the required memory of mobile visual recognition, we proposed a scalar-based classifier weight compression method before [6]. Although it is very simple and effective, it has the drawback that the performance is degraded largely in case of lower-bit representation. Then, in this paper, we propose an improved method to make 2-bit and 1-bit representation feasible, and make more comprehensive experiments including more large-scale 10k image classification with combination of the proposed improved scalar-based compression method and product quantization.

1. Introduction

Due to the recent progress of smartphone technologies, smartphones such as iPhone and Android phones have obtained much computational power which is almost comparable to consumer PCs. A quad-core CPU has lately become common to many smartphones. Taking advantage of their enhanced computation power, client-side image classification applications have been developed such as Impala [1], Jetpack Spotter [2] and FoodCam [5]. In mobile devices, the size of required memory of an application should be as small as possible, because the size of storage as well as the size of memory are limited. In addition, if large-scale visual recognition with small memory is made possible, it can be embedded into various kinds of electronic devices such cars, televisions and digital cameras, which helps make visual recognition technology more practical.

In this paper, we propose an improvement method of theirs to make 2-bit and 1-bit representation practical, and make more comprehensive experiments to show that the effectiveness of the proposed improved method is applicable for more wide-ranging visual classification problems. We will prove its effectiveness for more large-scale image classification on ImageNet 10k categories.

[1] http://www.euvt.eu/ (recently acquired by Qualcomm)
[2] https://www.jetpac.com/spotter/ (recently acquired by Google)

2. Related Work

In this section, we describe some related works on compression of Fisher Vector and mobile image recognition.

One of the reasons on high performance of FV compared to low dimensional features such as bag-of-features is its high dimensionality, which enhances its discriminative power. However, as a negative point on high dimensionality, it requires a large amount of memory and storage to store not only feature vectors but also weight vectors of trained linear classifiers. Therefore, in case of large-scale image classification where the number of training samples is large, to train a classifier with FV, batch learning of classifiers such as standard SVM is sometimes impossible, because it requires too much memory to store all the training samples. 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. Even in case of using an online learning method, Sanchez *et al.* [8] compressed feature vectors coded by FV with Product Quantization (PQ) [3] in training time to store a large number of feature vectors on memory at once. Zhang et al. [9] proposed mutual information based element selection of FV to reduce the dimension of FV and combined it with FV binarization proposed by Perronninn *et al.* [7]. However, they made experiments on only 262,144-d FV which is extremely high dimensional, and its effectives for lower dimensional FV (e.g. several thousands dim.) is unclear. In this way, compression of FV-coded feature vectors in training time for saving required memory has addressed so far.

On the other hand, memory saving for classification time which is our objective in this paper has not been explored before for image classification as long as we know except

in our previous work [6].

3. Overview of the Mobile Image Classification Pipeline

In this work, we adopt the framework with Fisher Vector and linear classifier for mobile large-scale visual recognition which is more widely applicable to various kinds of image classification problems in mobile environments than Deep Convolutional Neural Network (DCNN) at the present. In this section, we describe the pipeline on mobile visual recognition which is following [6].

Firstly, we extract RootHOG patches and Color patches in a dense grid sampling manner. Then, we apply PCA to all the extracted local features, and encode them into Fisher Vectors. Next, we evaluate linear classifiers in the one-vs-rest way by calculating dot-product FVs and quantized weights. Finally we output the top-N categories in terms of the descending order of evaluation scores of all the linear classifiers. The key point is using quantization of high-dimensional linear classifier weights for reducing the amount of required memory. In the experiments, it turns out that this reduction hardly affects classification accuracy.

3.1. Linear Classifier

As a method for mobile large-scale image classification, we adopt linear classifiers with one-vs-rest strategy. Akata *et al.* [1] examined various learning methods for a large-scale visual recognition including 10k ImageNet dataset. They claimed that the one-vs-rest strategy with online learning is recommended even for 10k-class classification as well as 1000-class classification.

In large-scale image classification, online learning is commonly used instead of batch learning. As an online learning method of linear classifiers, we use AROW [2] which is robust to noisy labels. This characteristics is suitable for a large-scale data, especially data gathered from the Web such as image data on ImageNet.

Classification with a weight vector estimated by AROW is just a calculation of dot-product between a FV-coded feature vector of the given image, $\mathbf{x}$, and the estimated weight, $\mathbf{w}$, regarding each of all the classes as follows.

$$S = \mathbf{w} \cdot \mathbf{x} \tag{1}$$

where S is an evaluation value of a linear classifier. $\mathbf{w}$ is the target we would like to compress in this work. Finally, we obtain the top-k categories in terms of the descending order of the evaluation scores of all the linear classifiers, since we adopt one-vs-rest strategy for multi-class classification. Note that although Eq.1 assume a linear classifier with only a weight vector, it is applicable for linear SVM, $S = \mathbf{w} \cdot \mathbf{x} + b$, which has bias b by regarding $[\mathbf{w};\ b]$ as a weight and $[\mathbf{x};\ 1]$ as a feature vector.

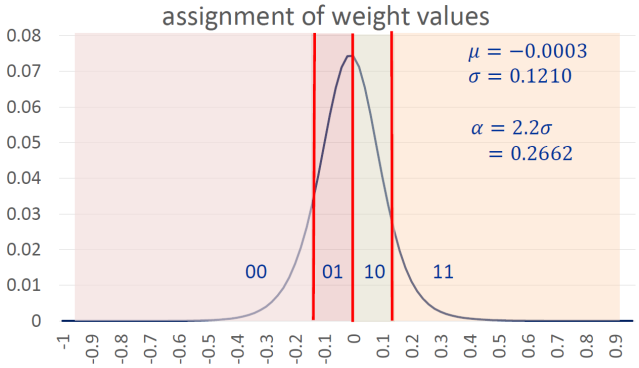


Figure 1. The distribution of all the elements of the weight vectors trained by AROW with the ILSVRC2012 dataset for FV of RootHOG patches and FV of Color patches in case of 64 GMM.

4. Quantization of Weight Vectors and its Improvement

Large-scale image classification with one-vs-rest linear classifiers and high dimensional Fisher Vector requires a large number of classifier weights. It is a problem for implementation on a mobile device. To save the amount of the required memory, we can reduce the dimension of FV with PCA or other projection methods. However, it will bring severe performance reduction [8]. To keep high classification performance, dimension reduction should be avoided. Note that mutual information based element selection of FV proposed by Zhang et al. [9] might have potential to reduce the dimension without severe performance loss, and to explore the combination of this work and their work is for future work.

Instead of dimension reduction of feature vectors, in this paper, we adopt a scalar-based quantization method for weight vectors of linear classifiers we proposed before [6]. In the method, we quantize each element of weight vectors without their dimension reduction. Because the method is based on scalar quantization, it has a good characteristic that we do not need to reconstruct quantized vectors into original ones when evaluating linear classifiers. This is different from the case of using generic compression methods, vector-quantization or product quantization (PQ) [3], which needs reconstruction of original vectors for evaluating classifiers.

In general, the element values of weight vectors of linear classifiers are distributed around zero. Figure 1 shows the distribution of weight element values estimated by AROW with the ILSVRC2012 dataset for both FV of Color patches and FV of RootHOG patches over 1000 classes in case of 64-GMM. The average and standard deviation of classifier weight elements are -0.0003 ± 0.1210. The shape of the distribution is exactly like a Gaussian distribution.

Based on this observation, as a first step, in this paper, we propose to restrict the range of an element value w_i within

$[-\alpha, \alpha)$ after subtracting w_{avg} where α is a positive constant value and w_{avg} is the average of all of the element values of all of the trained weight vectors, $w_i (i = 1..C)$, of C one-vs-rest linear classifiers. The important thing is that α and w_{avg} should be common through all of the linear classifiers in the one-vs-rest strategy. This enables us to avoid uncompression of the compressed weights in the evaluation time. It is represented by the following rules:

$$w'_i = (w_i - w_{avg})/\alpha \tag{2}$$

$$w''_i = \begin{cases} 0.999999 & (w'_i \geq 1) \quad (3a) \\ w'_i & (-1 < w'_i < 1) \quad (3b) \\ -1 & (w'_i \leq -1) \quad (3c) \end{cases}$$

In the previous method [6], we restricted the fixed range within $[-1, 1)$ and did not subtract the average value, which means it is assumed that the distribution and the average of the elements of the linear classifier weight vectors are almost the unchanged and zero. However, this caused the large performance degradation on lower bit representation, since this assumption is not always true.

Next, we quantize w''_i into n-bit representation with the following equation:

$$w'''_i = \lfloor w''_i \times 2^{n-1} + 2^{n-1} \rfloor, \tag{4}$$

where $\lfloor x \rfloor$ is a *floor* function representing 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. We use $w'''_i \ (i = 1..D)$ represented by a n-bit integer value as an element value of quantized weight vectors, where D represents the dimension size of a feature vector.

Regarding a constant value to decide the value range, α, in the experiments we decide α based on the standard deviation σ of the distribution of weight values of all the linear classifiers for one-vs-rest classification. From the preliminary experiments, we found α should be set as a value from 2σ to 3σ. In the experiments, we selected the best value of α from 2σ to 3σ by using validation data in the ILSVRC dataset. For the ILSVRC 1000-class data we set α as 2.2σ, while for the 100-class food image data we set α as 2.5σ.

Figure 1 shows an example of 2-bit quantization where the value range is divided into 4 2-bit states, 00, 01, 10 and 11 by setting the boundaries as $-2.2\sigma/2 (= -0.1331)$, 0 and $2.2\sigma/2$, respectively.

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, if we use the same α and w_{avg} for compression of all the weights of the one-vs-rest linear classifiers, we can omit to reconstruct original values. It is enough for classification to compute a dot-product between a scalar-quantized weight vector and a FV-coded feature vector as represented in Eq.1. Even reconstruction of the sign of the unsigned quantized values is not needed, because the constant values added to make the values unsigned can be ignored for relative comparison of the output values of the linear classifiers in the one-vs-rest classification. This prevents the processing time from increasing at the classification time on a smartphone. In fact, in case of 2-bit quantization shown in Figure 1, we assign just four kinds of the integer values, 0(00), 1(01), 2(10) and 3(11), to each of the value ranges as quantized values, respectively. In the experiments, surprisingly, no prominent performance loss was observed compared to the result with original floating value weights.

5. Experiments

In the experiments, firstly we used the ILSVRC2012 1000-class image dataset [4] for performance evaluation and evaluation of processing time on a smartphone, secondly we made experiments with ImageNet10k which contains 10,184 categories in the Fall 2009 release of ImageNet.

5.1. 1000-Class Large-Scale Classification

5.1.1 Experimental Setup

In the experiments, as a standard large-scale image dataset, we use the ILSVRC2012 dataset [4] 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 required memory and recognition speed as more important than performance.

First of all, we resize all the images so that the total pixels of all the image is less than 50,000 with the aspect ratio unchanged. Regarding local image features, we prepare two kinds of features, Color-patch, and RootHOG-patch, each of which represents color, and gradient, respectively. We sample them densely in every 5 pixels with two scales. Before being coded by Fisher Vector, they are applied with PCA and converted into 32-dim vectors in case of RootHOG-patch, and 24-dim vectors in case of Color-patch. Although 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, 128, and 256 Gaussians (64-GMM, 128-GMM, and 256-GMM in short) and Spatial Pyramid level 1 (1x1+2x2). As results, the total dimensions of FV are 10240, 20480, and 40960 for FV of RootHOG patches (RootHOG-FV) and 7680, 15360, and 30720 for FV of Color patches (Color-FV), respectively.

In classification time we trained RootHoG-FV and Color-FV independently, and we simply added two output

values of the linear classifiers on each class without any weighting. For weight quantization, we set α as 2.2σ which was estimated using validation data in the ILSVRC dataset as mentioned previously.

5.1.2 Evaluation on Weight Compression

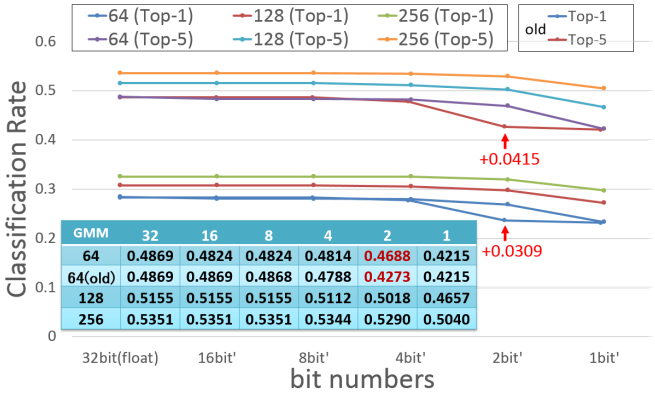
We compare the experimental results with GMM-64, GMM-128 and GMM-256 when varying the bits for weight compression from 1 to 32. In case that the number of bits is 32, we use the weight vectors represented by floating values, which means no compression.

Figure 2 shows the top-1 and top-5 classification rates for GMM-64, GMM-128, GMM-256 and the results from [6] which represented as "GMM-64(old)". From 32 bits to 4 bits, no prominent performance drops were observed. The rate was degraded by less than 1 points in case of 4-bit compression compared to the original floating weights (32bit) for all GMM sizes. In case of 2-bit, the slight performance drops were observed by 1.81%, 1.37% and 0.61% for GMM-64, GMM-128 and GMM-256, respectively, regarding the top-5 accuracy, while the large performance drops are observed by 5.96% for GMM-64(old). This indicates the performance loss is improved by 4.15% by the proposed method in the paper. In cases of 1-bit, the performance drops become prominent by more than abound 5 points for GMM-64 and GMM-128, and 3.1% for GMM-256. However, we can see that "GMM-256 with 1-bit" outperformed "GMM-128 with 2-bit", in both cases of which the required memory is the same. From these observation, performance loss due to weight quantization becomes smaller, as the size of the GMM becomes larger.

Then, we compare the classification performance for the three cases where the amount of the required memory to store the weight vectors is the same: "GMM-64 with 4-bit", "GMM-128 with 2-bit" and "GMM-256 with 1-bit (binary)". The result is shown in Figure 3. Among the three cases, "GMM-256 with 1-bit" was the best performance, 50.40%, and "GMM-128 with 2-bit" was almost comparable to the best, 50.18%. This results indicate that the weight vectors should be higher dimensional and lower-bit representation, when the available memory is fixed. However, for a smartphone implementation, we have to take into account processing time, especially, the time for Fisher Vector coding which is expected to be proportional to the GMM size. We will examine the relation between processing time and FV dimensions later.

5.1.3 Recognition Time

Figure 4 shows the processing times for 1000-class classification of one given image on Samsung Galaxy Note II in case of "64-GMM with 2-bits", "128-GMM with 2-bits"



GMM	32	16	8	4	2	1
64	0.4869	0.4824	0.4824	0.4814	0.4688	0.4215
64(old)	0.4869	0.4869	0.4868	0.4788	0.4273	0.4215
128	0.5155	0.5155	0.5155	0.5112	0.5018	0.4657
256	0.5351	0.5351	0.5351	0.5344	0.5290	0.5040

Figure 2. Top-1/Top-5 classification rates with GMM-64, GMM-128, GMM-256 and GMM-64(old) [6] with n-bit compressed weight vectors (n=1, 2, 4, 8, 16 and 32).

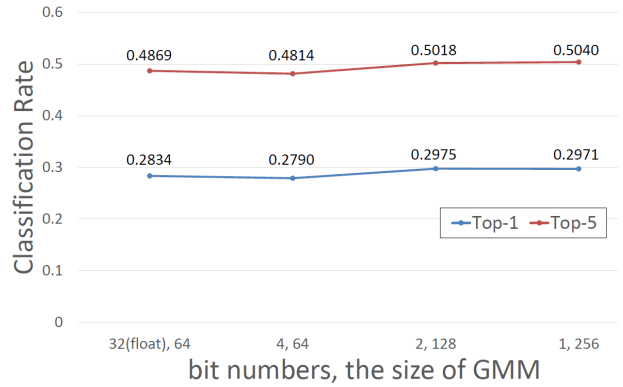


Figure 3. Comparison of Top-5 classification rate with three cases where the amount of the required memory to store the weight vectors is the same. Note that the results of "32(float, no quantization), GMM-64" is shown for reference.

and "256-GMM with 1-bit". "64-GMM with 2bits" has achieved the fastest time, 0.159 second. Although "256-GMM with 1-bit" achieved the best performance as shown in Figure 3, it was 3.7 times as slow as "GMM-64 with 2-bit". Since the performance difference between "GMM-128 with 2-bit" and "GMM-256 with 1-bit" is 0.22%, for a mobile implementation "GMM-128 with 2-bit" is currently the appropriate choice for the 1000-class ILSVRC dataset by taking account of the balance between accuracy and speed. However, when the CPU computational speed increases twice in the near future, "256-GMM with 1-bit" will become the better choice.

Basically the processing time consists of local feature extraction, applying PCA, FV coding, and evaluation of linear classifiers. The time for local feature extraction and computation of PCA is independent of the GMM size, while the processing time for FV coding and classifier evaluation is proportional to the GMM size. The time differences between GMM-64 and others are mostly caused by these GMM-size-dependent steps.

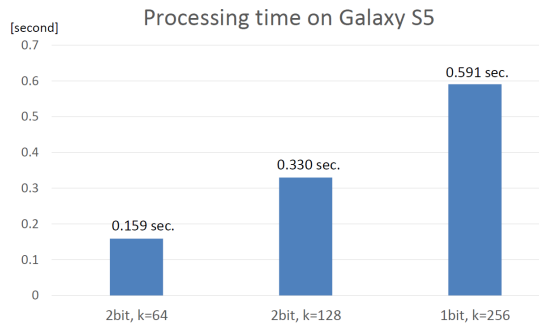


Figure 4. Processing time for 1000-class classification for one image on Samsung Galaxy S5 (2.6GHz, Quad-core, android 4.4.2).

5.2. More Large-Scale Dataset

We made experiments with ImageNet10k which contains 10,184 categories in the Fall 2009 release of ImageNet. We use 3072-dim FV of Color patch and 4120-dim FV of RootHOG with 128-GMM with no spatial pyramid as image features.

In this experiments, we compared the results with 4 different settings which are no compression, the proposed scalar-based compression with 2-bit, Product Quantization(PQ) [3], and PQ with PQ codebook compressed with scalar-based compression with 8-bit. For PQ, we vector-quantized every 8 elements of the weight vectors with 256 (8-bit) codewords, which leaded to 1-bit representation per element. Table 1 shows the results in top-1 and top-5 accuracy and the required memory size. While without compression the size of the total weight vectors, 292Mbyte, is too large for mobile implementation, with scalar compression the size, 18.2Mbytes, is feasible. Although the compression ratio for PQ is 1/32 which is half of the 2-bit scalar-based compression, the memory to store the codebooks is required to reconstruct original vectors for evaluation of classifiers. In this case, 7.3Mbytes is needed in addition to memory for PQ-coded vectors, and totally 16.4Mbytes is needed. Additionally, by applying the scalar-based method with 8-bit representation to PQ codebook, it can be reduced to 1.8Mbyte with only 0.09% performance loss, and total amount is 10.9MByte. This shows combination of PQ and the scalar-based compression is also effective.

Table 1. The results for ImageNet10k in case of no compression, the proposed scalar-based compression with 2-bit, Product Quantization(PQ) [3], and PQ with PQ codebook compressed with scalar-based compression with 8-bit. "PQCB" means the codebook of PQ.

method	top-1 (%)	top-5 (%)	memory (Mbyte)
no compression	11.83	25.25	292M
scalar	11.42	24.30	18.2M
PQ[3]	10.96	23.85	9.1M + 7.3M (PQCB)
PQ[3] + scalar	10.87	23.75	9.1M + 1.8M (8bit)

6. Conclusions and Future Work

In this paper, we proposed an improved scalar-based compression method for weight vectors of linear classifiers. With the method, we have implemented a client-side large-scale image classification on an Android smartphone, which can perform 1000-class classification for a given image in 0.159 seconds. In the experiments, we showed that compressing the weight vectors to 1/16 leaded to only 0.61% performance loss for 1000-class classification, which proved the effectiveness of the proposed method for large-scale classification on mobile devices and embedded devices where the amount of available memory is limited. In addition, the experimental results also proved that the proposed method enabled us to boost classification performance keeping memory size constant by compressing weights and instead increasing a codebook size even for normal-scale classification. When fixing the amount of memory for the weight vectors, the experimental result indicated that increasing the codebook size of Fisher Vector as much as possible and instead binarizing the weight vectors were the best choice in terms of classification performance. In other words, the gain by increasing the codebook size outperformed the loss by vectorization of the weight vectors. This is a new finding in this paper.

For future work, we will extend the proposed method to multi-layered deep neural network the structure of which is combination of linear classifiers and non-linear activation function such as ReLU, and implement the state-of-the-art large-scale image classifier.

References

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Good practice in large-scale learning for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):507–520, 2014. 2

[2] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. *Machine Learning*, 91(2):155–187, 2013. 2

[3] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. 1, 2, 5

[4] D. Jia, B. Alex, S. Sanjeev, S. Hao, K. Aditya, and L. Fei-Fei. Imagenet large scale visual recognition challenge 2012 (ILSVRC2012), 2012. http://www.image-net.org/challenges/LSVRC/2012/. 3

[5] Y. Kawano and K. Yanai. FoodCam: A real-time food recognition system on a smartphone. *Multimedia Tools and Applications*, 2014. 1

[6] Y. Kawano and K. Yanai. ILSVRC on a smartphone. *IPSJ Trans. on Computer Vision and Applications*, 6:83–87, 2014. 1, 2, 3, 4

[7] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *Proc. of IEEE Computer Vision and Pattern Recognition*, 2010. 1

[8] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *Proc. of IEEE Computer Vision and Pattern Recognition*, pages 1665–1672, 2011. 1, 2

[9] Y. Zhang, J. Wu, and J. Cai. Compact representation for image classification: To choose or to compress? In *Proc. of IEEE Computer Vision and Pattern Recognition*, 2014. 1, 2