Image Classification and Retrieval are ONE

Lingxi Xie[1], Richang Hong[2], Bo Zhang[3], and Qi Tian[4]
[1,3]LITS, TNLIST, Dept. of Computer Sci&Tech, Tsinghua University, Beijing 100084, China
[2]School of Computer and Information, Hefei University of Technology, Hefei 230009, China
[4]Department of Computer Science, University of Texas at San Antonio, TX 78249, USA
[1]198808xc@gmail.com,[2]hongrc@hfut.edu.cn,
[3]dcszb@mail.tsinghua.edu.cn,[4]qitian@cs.utsa.edu

ABSTRACT

In this paper, we demonstrate that the essentials of image classification and retrieval are the same, since both tasks could be tackled by measuring the similarity between images. To this end, we propose ONE (Online Nearest-neighbor Estimation), a unified algorithm for both image classification and retrieval. ONE is surprisingly simple, which only involves manual object definition, regional description and nearest-neighbor search. We take advantage of PCA and PQ approximation and GPU parallelization to scale our algorithm up to large-scale image search. Experimental results verify that ONE achieves state-of-the-art accuracy in a wide range of image classification and retrieval benchmarks.

Categories and Subject Descriptors

I.4.10 [**Image Processing and Computer Vision**]: Image Representation—*Statistical*; I.4.7 [**Image Processing and Computer Vision**]: Feature Measurement—*Feature representation*

General Terms

Algorithms, Experiments, Performance

Keywords

Image Classification; Image Retrieval; ONE; CNN

1. INTRODUCTION

Past decades have witnessed an impressive bloom of multimedia applications based on image understanding. For example, the number of categories in image classification has grown from a few to tens of thousands [13], and deep Convolutional Neural Networks (CNN) have been verified efficient in large-scale learning [25]. Meanwhile, image retrieval has been transplanted from toy programs to commercial search engines indexing billions of images, and new user intentions such as fine-grained concept search [62] are realized and proposed in this research field.

ICMR'15, June 23–26, 2015, Shanghai, China.

http://dx.doi.org/10.1145/2671188.2749289.

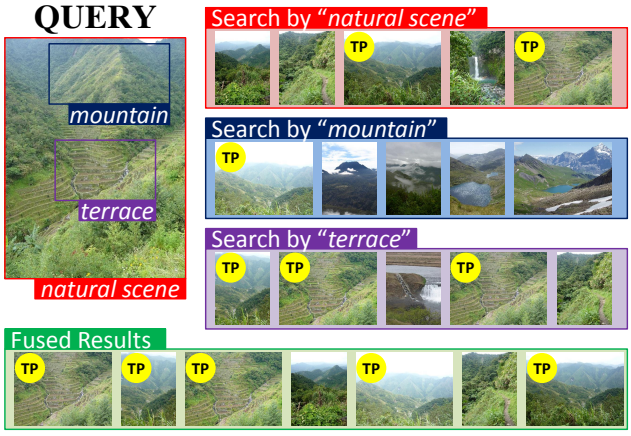

Figure 1: An image retrieval example illustrating the intuition of ONE (best viewed in color PDF). On a query image, it is possible to find a number of semantic objects. Searching for nearest neighbors with one object might not capture the exact query intention, but fusing yields satisfying results. A yellow circle with the word **TP** indicates a true-positive image. Images are collected from the **Holiday** dataset [20].

Both image classification and retrieval receive a **query** image at a time. Classification tasks aim at determining the class or category of the query, for which a number of training samples are provided and an extra training process is often required. For retrieval, the goal is to rank a large number of candidates according to their relevance to the query, and candidates are considered as independent units, *i.e.*, without explicit relationship between them. Both image classification and retrieval tasks could be tackled by the Bag-of-Visual-Words (BoVW) model. However, the ways of performing classification [10][26] and retrieval [46][38] are, most often, very different. Although all the above algorithms start from extracting patch or regional descriptors, the subsequent modules, including feature encoding, indexing/training and online querying, are almost distinct.

In this paper, we suggest using only ONE (Online Nearest-neighbor Estimation) algorithm for both image classification and retrieval. This is achieved by computing similarity between the query and each category or image candidate. Inspired by [4], we detect multiple **object proposals** on the query and each indexed image, and extract high-quality features on each object to provide better image description.

On the online querying stage, the query's relevance to a category or candidate image is estimated by the averaged nearest distance from querying objects to the objects in that category or candidate image. As shown in Figure 1, extracting more objects helps to find various visual clues and obtain better results. To improve efficiency, we leverage the idea of approximate nearest-neighbor search, and take advantage of GPU parallelization for fast computation. Experiments are performed on a wide range of image classification/retrieval datasets. Our algorithm achieves state-of-the-art accuracy with reasonable computational overheads.

The major contribution of this paper is summarized in three aspects. First, we reveal the possibility of unifying image classification and retrieval systems into ONE. Second, ONE achieves the state-of-the-art accuracy on a wide range of image classification and retrieval tasks, defending both training-free models for image recognition and regional features for near-duplicate object retrieval. Finally, we make full use of GPU parallelization to alleviate heavy online computational overheads, which might inspire various multimedia applications and research efforts in the future.

The remainder for this paper is organized as follows. Section 2 briefly introduces related works. Section 3 illustrates the ONE algorithm and key acceleration techniques. After showing experiments in Section 4, we conclude in Section 5.

2. RELATED WORKS

2.1 Image Classification

Image classification is a fundamental task which is aimed at categorizing images according to their semantic contents. Recent years, researchers propose to extend conventional tasks [26][16] in two aspects, *i.e.*, from coarse-grained to fine-grained [33][49][36], and from small-scale to large-scale [13].

The Bag-of-Visual-Words (BoVW) model is widely adopted to represent images with high-dimensional vectors. It often consists of three stages, *i.e.*, descriptor extraction, feature encoding and feature summarization. Due to the limited descriptive power of raw pixels, local descriptors such as SIFT [28][54] and HOG [11] are extracted. A visual vocabulary or codebook is then built to capture data distribution in the feature space. Descriptors are thereafter quantized onto the codebook as compact feature vectors [63][51][37][55], and summarized as an image-level representation [26][17][57][59]. These feature vectors are normalized [56], and fed into generalized machine learning algorithms [15][41] for training and testing. Besides, there are also efforts on designing training-free classification systems [4].

2.2 Image Retrieval

Image retrieval is closely related to a number of real-world multimedia applications. Given an image database and a query, it requires finding relative candidates to the query in a short time. As the rapid growth of the Internet [40], large-scale image search attracts more and more attentions in both academic and industrial fields [62].

The BoVW model is also widely adopted for image retrieval [34]. Handcrafted descriptors such as SIFT [28][58] and SURF [3] are extracted on the detected regions-of-interest (ROI) of an image [29][30]. A codebook, often with a large size, is trained in an approximate manner [38] to capture data distribution. Descriptors are quantized onto the codebook in an either hard [38] or soft manner [39]. Codebook training-free methods [73] are also suggested for feature encoding. Then the flowchart differs from classification in the way of organizing a huge number of features. An inverted index [46] is constructed to store the relationship between images and features. When a query image comes, descriptors are extracted and quantized accordingly to look up the inverted index. When the initial search results are available, they are often sent into a post-processing stage for higher precision and recall. Popular post-processing approaches include geometric verification [8], query expansion [9] and diffusion-based algorithms [23][60].

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are based on the theory that a multi-layer network is able to fit any complicated functions. In the early years, neural networks are verified efficient on simple recognition tasks [27]. Recent years, the available of large-scale training data (*e.g.*, ImageNet [13]) and powerful GPUs make it possible to train deep CNNs which significantly outperform the BoVW models [25].

A CNN is often composed of a number of layers. In each layer, responses from the previous layer are convoluted and activated by a differentiable function. Thus, the network could be formulated as a composed function, and the error signals produced by the difference between supervised and predicted labels could be back-propagated. Some recently proposed tricks are also crucial to help CNNs converge faster and prevent over-fitting, such as the ReLU activation function and the dropout technique [25]. It is suggested that deeper networks produce better recognition results [45][47], and intermediate responses of CNNs could also be transplanted onto other image applications [14][43]. The discussion of different configurations in CNNs is available in [6][65].

2.4 Approximate Nearest-neighbor Search

It is a common demand that finding the nearest neighbor of a D-dimensional query vector in a large database of N candidates. Since an exhaustive search algorithm requires as many as $O(ND)$ computations which is often intractable, approximate algorithms are adopted to accelerate this process with reasonable accuracy loss.

One family of solutions involve constructing a data structure for efficient lookup. Based on the k-d tree [18] or other similar structures [50], efficient Approximate Nearest Neighbor (ANN) search algorithm is proposed [31], and further applied on other applications such as large-scale vector clustering [38]. A notable drawback of such methods is the unavailability of distance between the query and each candidate, which makes it impossible to provide a ranklist for retrieval and related tasks.

There are also research efforts on encoding high-dimensional data into distance-preserving compact codes. A typical example is quantization, in which the distance between two vectors is estimated by the distance between codewords. Product Quantization (PQ) [21] trains a large codebook with the Cartesian product of small sub-codebooks. PQ then traverses each candidate and computes its distance to the query quickly with a fast pre-computation on codewords. Improvements of PQ include Cartesian K-Means [35] and Composite Quantization [69]. Another efficient approach, Locality Sensitive Hashing (LSH) [19], measures the distance by the Hamming distance and could be embedded to improve image search performance [70].

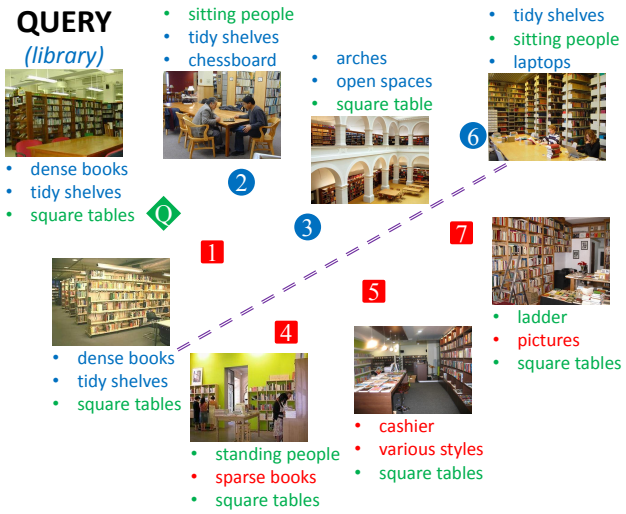

Figure 2: The difference between image classification and retrieval with the same query (green diamond). Red squares and blue circles indicate samples of the *bookstore* and *library* classes, respectively. For each image, three most significant visual attributes are listed, with the colors indicating their bias in visual concepts (red for *bookstore*, blue for *library*, and green for neutral). Candidates are ranked according to their relevance (distance) to the query. The dashed line illustrates the optimal linear classifier between two categories.

3. ALGORITHM

3.1 A Unified Framework

Since we aim at designing a unified framework for image classification and retrieval, we first make a brief discussion on the essential difference between these two tasks.

Both classification and retrieval involve measuring the similarity between the query and training or candidate images. The only difference lies in that a class label is provided for each training sample in classification. Therefore, we are actually computing the similarity between the query and a category, *i.e.*, a set of images. As observed in [4], image-to-class distance is much more stable than image-to-image distance. On the other hand, candidates in retrieval are often considered independently, *i.e.*, without explicit relationship between them. An intuitive example is shown in Figure 2. The query (a *library* image) is closest to a *bookstore* image in the feature space, and the outlier (#1) significantly harms the retrieval performance. However, when more labeled training samples are available, we obtain the optimal linear classifier between *library* and *bookstore* which predicts the correct label of the query.

In summary, classification tasks benefit from the class labels of images, but such information is not available in retrieval datasets. A direct solution is to augment the database by extracting multiple **object proposals** for each image, and consider each object as an individual image sample (with a "class label"). After each object is equipped with a high-quality regional descriptor, it is possible to deal with both image classification and retrieval problems by computing image-to-class distance [4].

3.2 Online Nearest-neighbor Estimation

This part formulates the previous ideas as the ONE (Online Nearest-neighbor Estimation) algorithm. We start from an image classification/retrieval dataset of N images,

$$\mathcal{I} = \{(\mathbf{I}_1, y_1), (\mathbf{I}_2, y_2), \ldots, (\mathbf{I}_N, y_N)\} \tag{1}$$

In which, $\mathbf{I}_n$ and y_n are the image data and label of the n-th image, respectively. For image classification, $y_n \in \{1, 2, \ldots, C\}$ is pre-defined by the dataset, while for image retrieval we simply set $C = N$ and $y_n = n$, indicating that each instance belongs to an independent "category".

For each image $\mathbf{I}_n$, a set of object proposals $\mathcal{P}_n$ is constructed,

$$\mathcal{P}_n = \{\mathbf{p}_{n,1}, \mathbf{p}_{n,2}, \ldots, \mathbf{p}_{n,K_n}\} \tag{2}$$

Here, $\mathbf{p}_{n,k} = \left(x^{\min}_{n,k}, y^{\min}_{n,k}, W_{n,k}, H_{n,k}, \theta_{n,k}\right)$ is the k-th object of the n-th image, denoted by the coordinate of its upper-left corner, its width and height, and $\theta_{n,k} \in [0^\circ, 360^\circ)$ indicating the angle by which the image is rotated before feature extraction. The designation of $\mathcal{P}_n$ will be discussed in detail later. Cropping and rotating $\mathbf{I}_n$ according to $\mathbf{p}_{n,k}$ yields a subimage $\mathbf{I}_{n,k}$, on which we compute a regional vector representation $\mathbf{f}_{n,k}$. Specifically, we extract 4096-D deep conv-net features which are the intermediate responses of a CNN [45]. Although other features such as VLAD [22] could also be adopted, we choose deep conv-net features because the excellent descriptive ability. According to a simple experiment based on NN feature search on the **Holiday** dataset [20], VLAD obtains a mAP score of 0.526 [20] while deep conv-net features get 0.642 [43].

Next we follow [4] to perform a Naive-Bayes Nearest-Neighbor (NBNN) search. We first define the feature set $\mathcal{F}_c$, $c = 1, 2, \ldots, C$, which is composed of all the features extracted from the objects that belong to the c-th category,

$$\mathcal{F}_c = \{\mathbf{f}_{n,k} \mid y_n = c \wedge 1 \leqslant k \leqslant K_n\} \tag{3}$$

For a query image $\mathbf{I}_0$, we compute its distance to each category $c \in \{1, 2, \ldots, C\}$, which is the averaged distance between each object of $\mathbf{I}_n$ and its nearest neighbor in $\mathcal{F}_c$,

$$\begin{aligned} \mathrm{dist}(\mathbf{I}_0, c) &\doteq \mathrm{dist}(\mathbf{I}_0, \mathcal{F}_c) && (4)\\ &= \frac{1}{K_0}\sum_{k=1}^{K_0} \mathrm{dist}(\mathbf{f}_{0,k}, \mathcal{F}_c) && (5)\\ &= \frac{1}{K_0}\sum_{k=1}^{K_0} \min_{\mathbf{f}\in\mathcal{F}_c} \|\mathbf{f}_{0,k} - \mathbf{f}\|_2^2 && (6) \end{aligned}$$

When all the image-to-category distances are available, it is convenient to analyze them for specified purposes, such as finding the minimum one for category prediction, or sorting them for a ranklist of candidates.

3.3 Object Proposals

It remains to construct an object proposal set $\mathcal{P}_n$ for each image $\mathbf{I}_n$. For this purpose, we might refer to unsupervised object detectors, including Objectness [1], Selective Search [48] and Binarized Normed Gradients (BING) [7]. These algorithms often provide a number of bounding boxes which are the possible locations of objects. By sorting the confidence scores of the boxes, it is possible to obtain an arbitrary number K_D of top-ranked object proposals[1].

[1]Due to the limited space, please refer to the publications above for examples of object detection.

In comparison, we also adopt an alternative approach which extracts manually defined objects on an image. For this, we first define the number of object proposal layers L_{O}, and then enforce that objects within one layer have the same size meanwhile are distributed as dispersive as possible. Denote the width and height of image $\mathbf{I}_n$ as W_n and H_n. In the l-th layer, there are $r_l \times r_l$ objects with a fixed size $\frac{W_n}{s_l} \times \frac{H_n}{s_l}$, where r_l and s_l are the object density and scale parameters which will be discussed later. The upper-left corner of the (a, b)-th box, $0 \leqslant a, b < r_l$, is located at $\left(a \cdot \left(W_n - \frac{W_n}{s_l}\right) / (r_l - 1), b \cdot \left(H_n - \frac{H_n}{s_l}\right) / (r_l - 1)\right)$. An L_{O}-layer model has $K_{L_{\mathrm{O}}} = \sum_{l=1}^{L_{\mathrm{O}}} r_l^2$ object proposals.

Either a detected or manually defined object might be rotated. Here we consider 4 simplest rotations, *i.e.*, $\theta_{n,k} \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$. As an increasing number of objects might result in heavier computational overheads, we might only rotate the top-K_{R} detected regions ($K_{\mathrm{R}} \leqslant K_{\mathrm{D}}$), or regions on the first L_{R} ($L_{\mathrm{R}} \leqslant L_{\mathrm{O}}$) layers. Such a strategy produces $K_{\mathrm{D}} + 3K_{\mathrm{R}}$ or $K_{L_{\mathrm{O}}} + 3K_{L_{\mathrm{R}}}$ object proposals.

The comparison between automatically detected and manually defined object proposals is similar to that between detected and densely sampled descriptors which is well discussed in the conventional Bag-of-Visual-Words (BoVW) models. We will show in Section 4.2 that both strategies obtain satisfying classification and retrieval results.

3.4 Approximate Nearest-neighbor Search

Since ONE requires a huge number of NN queries, linear exhaustive NN search might be computationally expensive. We adopt both Principal Component Analysis (PCA) and Product Quantization (PQ) [21] to reduce time complexity. Deep conv-net features [14] (4096 dimensions) of each region are first reduced to D dimensions by PCA. Then it is partitioned by PQ into M segments and each subvector is quantized onto one of T codewords.

It is worth noting that vector dimensions after PCA reduction are ranked by decreasing energy, whereas PQ works better in the case that each segment contains vectors with approximately equal information. Therefore, we perform dimension rearrangement to make PCA and PQ cooperate better, in which the m-th PQ segment is composed of the $(m, M + m, 2M + m, \ldots, D - M + m)$-th PCA-reduced dimensions. In practise, this strategy consistently boosts the classification/retrieval accuracy by about 2%. It is also instructive to whiten PCA features for similar effects.

Assume there are N training/candidate images in a classification/retrieval task. For each image, K object windows are proposed, on each of which we extract a 4096-dimensional feature. Therefore for each query, we need to process K queries in a database of KN vectors. An exhaustive NN search takes $O\big(4096K^2N\big)$ time and $O(4096KN)$ memory. In an approximate NN search, features are PCA-reduced to D dimensions, and quantized with a PQ of M segments and T codewords for each segment. According to PQ [21], the time complexity is $O\big(K^2NM + KDT\big)$, meanwhile storing quantized vectors and a codebook requires $KNM \log_2 T$ bits and DT floating numbers in total. Since $M \ll D \ll 4096$ and $T \ll N$, both time and memory costs are greatly reduced. When the parameters are fixed, the computational complexity grows linearly with the dataset size N, which guarantees that our algorithm scales up well.

Section 4.4 provides time/memory costs in experiments.

3.5 GPU Acceleration

Despite the approximation by PCA and PQ, the computational cost on the online querying stage is still very high, *e.g.*, about 5×10^{11} floating operations to search one query among one million candidates (see Section 4.4). Fortunately, most of the heavy computation comes from the linear search stage of PQ. We can take advantage of GPU parallelization for efficient acceleration.

Graphics Processing Unit (GPU) is a specialized electronic device designed to rapidly manipulate and alter memory to accelerate image processing in a frame buffer intended for output to a display. It often contains a large number of stream processors in which large data blocks are processed in parallel. The highly parallel structure of GPUs makes them more effective than CPUs for large-scale simple-arithmetic operations. Recent years, GPUs have been widely adopted for accelerating deep CNN training [25].

It is worth noting that the storage of a GPU is often limited, *e.g.*, an NVIDIA GeForce GTX Titan has only 6GB memory. We shall carefully design the parameters discussed above, so that quantized vectors could be stored in a GPU. An extensive study on the proper parameters are provided in Section 4.2.

4. EXPERIMENTS

4.1 Datasets and Implementation Details

For image classification, we use the **LandUse-21** dataset [64], the **Indoor-67** dataset [42] and the **SUN-397** dataset [53] for scene classification. 80, 80 and 50 images per category are randomly selected for training. We also use the Oxford **Pet-37** dataset [36], the Oxford **Flower-102** dataset [33] and the Caltech-UCSD **Bird-200**-2011 dataset [49] for fine-grained object recognition. 100, 20 and 30 images per category are randomly selected for training. The **SUN-397** dataset [53] is one of the largest available datasets for scene understanding, which provides an evidence that our algorithm scales up well. For each dataset, the random training/testing splits are repeated for 10 rounds and averaged classification accuracy is reported.

For image retrieval, we use the **UKBench** dataset [34] and the **Holiday** dataset [20]. They are both near-duplicate instance search datasets. The **UKBench** dataset consists of 2550 objects and 4 photos are taken for each of them (10200 images in total). The **Holiday** dataset is composed of 500 groups of objects or scenes with a handful of instances in each of them. The standard evaluation uses the N-S score (the average number of true-positive images in top-4 results) for **UKBench**, and the mAP (mean average precision) score for **Holiday**. To test the scalability of our model, we also mix the **Holiday** dataset with one million irrelevant images crawled from the Web.

For manually defined objects, we use $L_{\mathrm{O}} = 5$ layers, with fixed parameters $(s_1, s_2, s_3, s_4, s_5) = (1.0, 1.2, 1.5, 2.0, 2.5)$ and $(r_1, r_2, r_3, r_4, r_5) = (1, 2, 3, 4, 5)$, respectively. For automatic object detection, we use Selective Search [48] and choose top-ranked objects according to the confidence scores. For object representation, we compute a pre-trained 19-layer **VGG-Net** [45], and extract the 4096-D responses of the second-to-last fully connected layers, without applying ReLU activation. Feature vectors are square-root normalized and then ℓ_2-normalized after PCA reduction.

Objects	$L_O = 1$	$L_O = 2$	$L_O = 3$	$L_O = 4$	$L_O = 5$
$L_R = 0$	67.18	73.41	77.63	82.18	85.71
$L_R = 1$	67.40	73.69	77.95	82.42	86.11
$L_R = 2$	–	73.78	78.01	82.47	86.14
$L_R = 3$	–	–	78.03	82.51	86.18
$L_R = 4$	–	–	–	82.53	86.22
$L_R = 5$	–	–	–	–	**86.24**

(a) Classification Accuracy (%) on **Flower-102**

Objects	$L_O = 1$	$L_O = 2$	$L_O = 3$	$L_O = 4$	$L_O = 5$
$L_R = 0$	0.751	0.772	0.796	0.826	0.847
$L_R = 1$	0.816	0.832	0.837	0.854	0.871
$L_R = 2$	–	0.838	0.848	0.858	0.874
$L_R = 3$	–	–	0.861	0.868	0.880
$L_R = 4$	–	–	–	0.882	0.883
$L_R = 5$	–	–	–	–	**0.887**

(b) Retrieval Accuracy (mAP) on **Holiday**

Table 1: Classification and retrieval accuracy with respect to the complexity of manually defined objects.

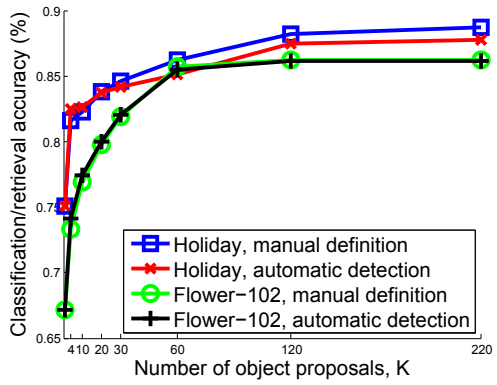

Figure 3: Classification/retrieval accuracy with respect to the way of defining objects and the number of proposals. We first construct $K = 220$ objects ($L_O = L_R = 5$ for manual definition and $K_D = K_R = 55$ for automatic detection), and randomly select a subset of them for evaluation.

4.2 Model and Parameters

We discuss the parameters of the ONE algorithm, *i.e.*, the object proposal set $\mathcal{P}$, the PCA-reduced dimension D, and the number of segments M and codewords T for PQ. For quick evaluations, we evaluate the classification and retrieval accuracy on two relatively smaller datasets, namely the **Flower-102** dataset and the **Holiday** dataset.

We first use accurate NN search to evaluate classification and retrieval accuracy with respect to the object proposals. Results are summarized in Table 1 (the number of layers of manually defined object proposals) and Figure 3 (automatic detection vs. manual definition with random selection), respectively. One can observe that a larger number of object proposals often produce better accuracy. Meanwhile, manually defined and automatically detected objects produce comparable results, suggesting that it is the number

	LandUse-21	**Indoor-67**	**SUN-397**
[24]	92.8	63.4	46.1
[61]	–	63.48	45.91
[14]	–	–	40.94
[43]	–	69.0	–
SVM	**94.52**	68.46	53.00
ONE	93.98	**69.61**	**54.47**
ONE+SVM	**94.71**	**70.13**	**54.87**

(a) Scene Recognition Accuracy (%)

	Pet-37	**Flower-102**	**Bird-200**
[2]	54.30	80.66	–
[52]	59.29	75.26	–
[32]	56.8	84.6	33.3
[14]	–	–	58.75
[43]	–	86.8	61.8
SVM	88.05	85.49	59.66
ONE	**89.50**	**86.24**	**61.54**
ONE+SVM	**90.03**	**86.82**	**62.02**

(b) Fine-Grained Recognition Accuracy (%)

Table 2: Classification accuracy on different datasets. In each subtable, the first and second parts contain algorithms without and with using deep conv-net features, respectively.

of object proposals that makes the major contribution to boosting accuracy. When the number of object proposals is sufficiently large (*e.g.*, $K \geqslant 60$), manually defined objects work even slightly better (86.24% vs. 86.16% on **Flower-102** and 0.887 vs. 0.878 on **Holiday**). Therefore, we only use manually defined objects in later experiments.

For large-scale image search, we consider both PCA and PQ for acceleration (see Section 3.4). The impact of parameters, *i.e.*, PCA dimension D, PQ segments M and codewords T, are summarized in Figure 4, 5 and 6, respectively. We use $K = 220$ object proposals ($L_O = L_R = 5$) in these experiments. Empirically, a proper set of parameters achieve a good tradeoff between accuracy and computational overheads. For example, partitioning a vector into 128 segments produces nearly the same accuracy compared to 64 segments, but requires almost doubled time and memory consumption on the online querying stage. We choose parameters $D = 512$, $M = 32$ and $T = 4096$ as an accuracy-complexity tradeoff. For small-scale experiments, *i.e.*, the number of training or candidate images is not greater than 10^4, we use PCA but do not use PQ for higher classification and retrieval accuracy.

Moreover, we need to constraint the number of object proposals K to fit the limited GPU memory. Recall from Section 3.4 that PQ stores $KNM \log_2 T$ bits and DT floating numbers, *i.e*, $\frac{1}{8}KNM \log_2 T + 4DT$ bytes, which should be less than 6GB (the memory of a Titan GPU). When we are dealing with one million images ($N \approx 10^6$) in large-scale experiments, K shall be no larger than 120. We use $L_O = L_R = 4$ ($K = 120$) in practise. Of course the tradeoff between more objects and more rotations could be determined according to detailed conditions. For example, in the **Oxford Buildings** dataset [38] (less image rotation), it might be better to use $L_O = 6$ and $L_R = 2$ ($K = 106$).

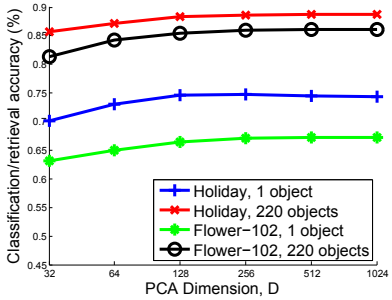

Figure 4: Accuracy w.r.t. PCA dimension D. PQ is not used here.

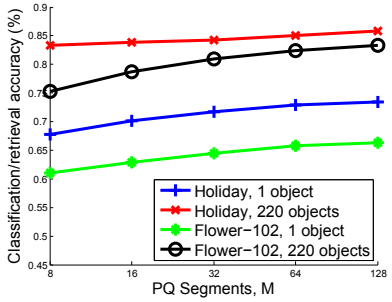

Figure 5: Accuracy w.r.t. PQ segments M, with $D = 1024$ and $T = 4096$.

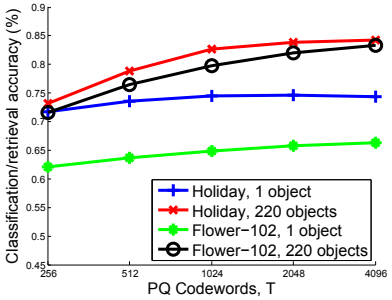

Figure 6: Accuracy w.r.t. PQ codewords T, with $D = 1024$ and $M = 32$.

	BoVW+ONE	ONE	BoVW	[70]	[43]	[71]	[12]	[67]	[68]
Holiday	**0.899**	**0.887**	0.518	0.881	0.843	0.858	0.847	0.846	0.809
UKBench	**3.887**	**3.873**	3.134	3.873	–	3.85	3.75	3.77	3.60

Table 3: Comparison of retrieval accuracy with the state-of-the-arts. Among the competitors, [70] and [43] also use deep conv-net features extracted on the **Alex-Net** [25]). While [70], [12] and [67] adopt post-processing, ONE does not.

4.3 Comparison to the State-of-the-Arts

We compare ONE with the state-of-the-art algorithms.

Image classification results are summarized in Table 2. One could observe that our algorithm achieves competitive accuracy in both scene recognition and fine-grained recognition tasks. For fine-grained recognition, we do not use any part detectors but simply extract regular object proposals, leading to inferior classification accuracy compared to those using complicated part detectors, such as [66] (73.89%) and [5] (85.4%) on the **Bird-200** dataset. Our algorithm could also cooperate with specialized part detectors.

Since a training process provides much benefit for image classification, we also compare our model with conventional machine learning approaches. We adopt LibLINEAR [15], a scalable SVM implementation with a tradeoff parameter $C = 10$. In most cases, ONE outperforms SVM, defending the ability of a training-free algorithm for classification. The fusion of ONE and SVM results, obtained by adding the confidence scores directly and re-ranking, also shows superior performance over both models, suggesting that complementary information is captured by different approaches.

Next we report image retrieval accuracy in Table 3. One could observe that our algorithm, without post-processing, outperforms all the competitors. Similar to image classification, we also compare ONE with the conventional BoVW model. Our model is implemented with SIFT extraction, large codebook training and hard quantization [38], an inverted index structure and ℓ_p-norm weighting [72]. Although ONE outperforms BoVW significantly, we notice that BoVW works better to capture local clues that help retrieval. When we fuse the results generated by both models, they complement each other and produce even higher accuracy. To the best of our knowledge, the 0.899 mAP score on **Holiday** and the 3.887 N-S score on **UKBench** rank among the highest scores ever reported on these two datasets. When **Holiday** is mixed with one million distractors, we get a 0.758 mAP score with PCA and PQ approximation, which outperforms 0.724 reported in [70] and 0.633 in [68].

The excellent classification and retrieval performance comes from a perfect cooperation of object detection and description, *i.e.*, efficient image representation by conv-net features and powerful retrieval process by ONE. Either applying NN search on global image features (see Table 1) or replacing deep conv-net features with BoVW-based features would case dramatic accuracy drop.

4.4 Time and Memory Costs

This part provides an experimental study on the computational overheads of ONE. A theoretical analysis could be found in Section 3.4. Parameters, *i.e.*, $D = 512$, $M = 32$ and $C = 4096$, are inherited from Section 4.2. Time and memory costs of different models are summarized in Table 4.

For image classification, we take **SUN-397** [53], the largest dataset in our experiments, as the example. Following the same setting provided by the authors, the numbers of training and testing images in one split are both $N = 397 \times 50 \approx 20\text{K}$. $K = 220$ ($L_\text{O} = L_\text{R} = 5$) object proposals are extracted on each image. According to the analysis in Section 3.4, classifying a single image requires approximately 4.5×10^8 floating multiplications and 3.2×10^9 floating summations, which takes less than 0.1s on a Titan GPU, *i.e.*, about 1800s (0.5h) on the whole testing set. The memory storage on GPU is about 200MB in this case. Compared with conventional SVM classification which requires 8h and 2560MB for one training and testing, our method is significantly faster in time and cheaper in memory.

For image retrieval, we evaluate our algorithm on the **Holiday** dataset [20] with one million distractors. With $N \approx 1\text{M}$ and $K = 120$ ($L_\text{O} = L_\text{R} = 4$), we need approximately 2.4×10^8 floating multiplications and 4.5×10^{11} floating summations for each query, which require about 1.2s on a Titan GPU, which is comparable with conventional systems performed on CPU. The storage of about 45G bits and 2M floating numbers fits well in the 6GB GPU memory. Both time and memory costs are comparable to the state-of-the-art approaches with deep conv-net features [70].

	Classification with **ONE**	Classification with BoVW	Retrieval with **ONE**	Retrieval with BoVW
Descriptor Extraction Time (s, per image)	1.81 (CNN)	1.36 (SIFT)	1.75 (CNN)	0.83 (SIFT)
Codebook Training Time (h)	0.39 (PQ)	2.41 (GMM)	0.39 (PQ)	6.18 (AKM)
Codebook Training Memory (GB)	0.63	2.50	0.63	8.31
Feature Quantization Time (s, per image)	0.17 (PQ)	0.23 (FV)	0.17 (PQ)	0.10 (VQ)
Offline Training Time (h)	–	7.71 (SVM)	–	2.85 (IND)
Offline Training Memory (GB)	–	2.50 (SVM)	–	4.19 (IND)
Online Querying Time (s, per query)	0.08	< 0.01	1.17	0.56
Online Querying Memory (GB)	0.21 (PQ)	0.05	5.65 (PQ)	4.19 (IND)

Table 4: Comparison of computational costs between ONE and BoVW. Results are reported on **SUN-397** (about 100K images) for classification, and **Holiday** with 1M distractors for retrieval. For abbreviations: **GMM** – Gaussian Mixture Model, **AKM** – Approximate K-Means [38], **FV** – Fisher Vectors [44], **VQ** – Vector Quantization, **IND** – inverted index.

It is worth noting that the actual computational costs in ONE are much more expensive than conventional algorithms. For example, ONE requires nearly 5×10^{11} floating operations for searching one query among one million candidates, while a simple BoVW-based approach often needs no more than 10^9 [34][73]. GPU is the key to make ONE produce results in a reasonable time (*e.g.* around one second). Conventional algorithms often contain complicated modules (*e.g.*, inverted index, spatial verification, *etc.*) with asynchronous memory access and/or a large number of serial operations, making them difficult to be transplanted to GPU for acceleration. As the rapid development of multi-GPU, our algorithm might attract more attentions in the future.

5. CONCLUSIONS

This paper proposes to unify image classification and retrieval algorithms into ONE (Online Nearest-neighbor Estimation). We demonstrate that, with the help of high-quality regional features, both classification and retrieval tasks could be accomplished with a simple NBNN search [4]. We take advantage of PCA and PQ approximation as well as GPU parallelization to alleviate heavy computational costs. Despite the simplicity, our algorithm achieves state-of-the-art image classification and retrieval performance.

The success of our algorithm inspires future research in three aspects. First, the essence of image classification and retrieval are the same, both of them could be tackled by measuring image similarity. Second, extracting more objects often leads to higher classification/retrieval accuracy, enlightening that image description, even with deep convnet features, is far from complete. Third, GPU is the future of high performance computing, therefore designing a GPU-friendly algorithm is necessary and beneficial.

6. ACKNOWLEDGMENTS

This work was supported by the 973 Program of China (Grant Nos. 2013CB329403 and 2012CB316301), the NNSF of China (Grant Nos. 61332007, 61273023 and 61429201), and the Tsinghua University Initiative Scientific Research Program (Grant No. 20121088071). This work was also supported in part to Dr. Hong by State High-Tech Development Plan 2014AA015104, and the Program for New Century Excellent Talents in University under grant NCET-13-0764; and in part to Dr. Tian by ARO grant W911NF-12-1-0057, and Faculty Research Awards by NEC Labs of America.

7. REFERENCES

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the Objectness of Image Windows. *TPAMI*, 2012.

[2] A. Angelova and S. Zhu. Efficient Object Detection and Segmentation for Fine-Grained Recognition. *CVPR*, 2013.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. *ECCV*, 2006.

[4] O. Boiman, E. Shechtman, and M. Irani. In Defense of Nearest-Neighbor Based Image Classification. *CVPR*, 2008.

[5] S. Branson, G. Van Horn, S. Belongie, and P. Perona. Bird Species Categorization Using Pose Normalized Deep Convolutional Nets. *arXiv preprint arXiv:1406.2952*, 2014.

[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. *BMVC*, 2014.

[7] M. Cheng, Z. Zhang, W. Lin, and P. Torr. BING: Binarized Normed Gradients for Objectness Estimation at 300fps. *CVPR*, 2014.

[8] O. Chum, M. Perdoch, and J. Matas. Geometric Min-Hashing: Finding a (Thick) Needle in a Haystack. *CVPR*, 2009.

[9] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval. *ICCV*, 2007.

[10] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual Categorization with Bags of Keypoints. *ECCV*, 2004.

[11] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. *CVPR*, 2005.

[12] C. Deng, R. Ji, W. Liu, D. Tao, and X. Gao. Visual Reranking through Weakly Supervised Multi-Graph Learning. *ICCV*, 2013.

[13] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A Large-scale Hierarchical Image Database. *CVPR*, 2009.

[14] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *ICML*, 2014.

[15] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A Library for Large Linear Classification. *JMLR*, 2008.

[16] L. Fei-Fei, R. Fergus, and P. Perona. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. *CVIU*, 2007.

[17] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric Lp-norm Feature Pooling for Image Classification. *CVPR*, 2011.

[18] J. Friedman, J. Bentley, and R. Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM ToMS*, 1977.

[19] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. *VLDB*, 1999.

[20] H. Jegou, M. Douze, and C. Schmid. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. *ECCV*, 2008.

[21] H. Jegou, M. Douze, and C. Schmid. Product Quantization for Nearest Neighbor Search. *TPAMI*, 2011.

[22] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating Local Descriptors into a Compact Image Representation. *CVPR*, 2010.

[23] Y. Jing and S. Baluja. VisualRank: Applying PageRank to Large-Scale Image Search. *TPAMI*, 2008.

[24] T. Kobayashi. Dirichlet-based Histogram Feature Transform for Image Classification. *CVPR*, 2014.

[25] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*, 2012.

[26] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *CVPR*, 2006.

[27] B. LeCun, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten Digit Recognition with a Back-Propagation Network. *NIPS*, 1990.

[28] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 2004.

[29] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions. *Image and Vision Computing*, 2004.

[30] K. Mikolajczyk and C. Schmid. Scale & Affine Invariant Interest Point Detectors. *IJCV*, 2004.

[31] M. Muja and D. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *VISIGRAPP*, 2009.

[32] N. Murray and F. Perronnin. Generalized Max Pooling. *CVPR*, 2014.

[33] M. Nilsback and A. Zisserman. Automated Flower Classification over a Large Number of Classes. *Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

[34] D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. *CVPR*, 2006.

[35] M. Norouzi and D. Fleet. Cartesian K-Means. *CVPR*, 2013.

[36] O. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. Cats and Dogs. *CVPR*, 2012.

[37] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher Kernel for Large-scale Image Classification. *ECCV*, 2010.

[38] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. *CVPR*, 2007.

[39] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases. *CVPR*, 2008.

[40] G. Qi, C. Aggarwal, Q. Tian, H. Ji, and T. Huang. Exploring Context and Content Links in Social Media: A Latent Space Method. *TPAMI*, 2012.

[41] G. Qi, X. Hua, Y. Rui, J. Tang, and H. Zhang. Two-Dimensional Multi-Label Active Learning with an Efficient Online Adaptation Model for Image Classification. *TPAMI*, 2009.

[42] A. Quattoni and A. Torralba. Recognizing Indoor Scenes. *CVPR*, 2009.

[43] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN Features off-the-shelf: an Astounding Baseline for Recognition. *CVPR*, 2014.

[44] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image Classification with the Fisher Vector: Theory and Practice. *IJCV*, 2013.

[45] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[46] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. *ICCV*, 2003.

[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. *NIPS*, 2014.

[48] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective Search for Object Recognition. *IJCV*, 2013.

[49] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. *Technical Report*, 2011.

[50] J. Wang, N. Wang, Y. Jia, J. Li, G. Zeng, H. Zha, and X. Hua. Trinary-Projection Trees for Approximate Nearest Neighbor Search. *TPAMI*, 2013.

[51] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-Constrained Linear Coding for Image Classification. *CVPR*, 2010.

[52] Z. Wang, J. Feng, and S. Yan. Collaborative Linear Coding for Robust Image Classification. *IJCV*, 2014.

[53] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun Database: Large-scale Scene Recognition from Abbey to Zoo. *CVPR*, 2010.

[54] L. Xie, Q. Tian, J. Wang, and B. Zhang. Image Classification with Max-SIFT Descriptors. *ICASSP*, 2015.

[55] L. Xie, Q. Tian, M. Wang, and B. Zhang. Spatial Pooling of Heterogeneous Features for Image Classification. *TIP*, 2014.

[56] L. Xie, Q. Tian, and B. Zhang. Feature Normalization for Part-based Image Classification. *ICIP*, 2013.

[57] L. Xie, Q. Tian, and B. Zhang. Hierarchical Part Matching for Fine-Grained Visual Categorization. *ICCV*, 2013.

[58] L. Xie, Q. Tian, and B. Zhang. Max-SIFT: Flipping Invariant Descriptors for Web Logo Search. *ICIP*, 2014.

[59] L. Xie, Q. Tian, and B. Zhang. Generalized Regular Spatial Pooling for Image Classification. *ICASSP*, 2015.

[60] L. Xie, Q. Tian, W. Zhou, and B. Zhang. Fast and Accurate Near-Duplicate Scale Web Image Search via Graph Propagation and Search Process Tradeoff. *CVIU*, 2014.

[61] L. Xie, J. Wang, B. Guo, B. Zhang, and Q. Tian. Orientational Pyramid Matching for Recognizing Indoor Scenes. *CVPR*, 2014.

[62] L. Xie, J. Wang, B. Zhang, and Q. Tian. Fine-Grained Image Search. *TMM*, 2015.

[63] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification. *CVPR*, 2009.

[64] Y. Yang and S. Newsam. Bag-of-Visual-Words and Spatial Extensions for Land-Use Classification. *ICAGIS*, 2010.

[65] M. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. *ECCV*, 2014.

[66] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-Based R-CNNs for Fine-Grained Category Detection. *ECCV*, 2014.

[67] S. Zhang, M. Yang, T. Cour, K. Yu, and D. Metaxas. Query Specific Fusion for Image Retrieval. *ECCV*, 2012.

[68] S. Zhang, M. Yang, X. Wang, Y. Lin, and Q. Tian. Semantic-aware Co-indexing for Image Retrieval. *ICCV*, 2013.

[69] T. Zhang, C. Du, and J. Wang. Composite Quantization for Approximate Nearest Neighbor Search. *ICML*, 2014.

[70] L. Zheng, S. Wang, F. He, and Q. Tian. Seeing the Big Picture: Deep Embedding with Contextual Evidences. *arXiv preprint arXiv:1406.0132*, 2014.

[71] L. Zheng, S. Wang, Z. Liu, and Q. Tian. Packing and Padding: Coupled Multi-Index for Accurate Image Retrieval. *CVPR*, 2014.

[72] L. Zheng, S. Wang, and Q. Tian. Lp-Norm IDF for Scalable Image Retrieval. *TIP*, 2014.

[73] W. Zhou, Y. Lu, H. Li, and Q. Tian. Scalar Quantization for Large Scale Image Search. *ACM Multimedia*, 2012.