Compression of Deep Neural Networks for Image Instance Retrieval

Vijay Chandrasekhar[1,4], Jie Lin[1], Qianli Liao[3],
Olivier Morère[2], Antoine Veillard[2], Lingyu Duan[5], Tomaso Poggio[3]
[1]*Institute for Infocomm Research, Singapore*
[2] *Université Pierre et Marie Curie, Paris, France*
[3] *Massachusetts Institute of Technology, Boston, MA*
[4] *Nanyang Technological University, Singapore*
[5] *Peking University, China*

Abstract

Image instance retrieval is the problem of retrieving images from a database which contain the same object. Convolutional Neural Network (CNN) based descriptors are becoming the dominant approach for generating global image descriptors for the instance retrieval problem. One major drawback of CNN-based global descriptors is that uncompressed deep neural network models require hundreds of megabytes of storage making them inconvenient to deploy in mobile applications or in custom hardware. In this work, we study the problem of neural network model compression focusing on the image instance retrieval task. We study quantization, coding, pruning and weight sharing techniques for reducing model size for the instance retrieval problem. We provide extensive experimental results on the trade-off between retrieval performance and model size for different types of networks on several data sets providing the most comprehensive study on this topic. We compress models to the order of a few MBs: two orders of magnitude smaller than the uncompressed models while achieving negligible loss in retrieval performance. [1]

1. Introduction

Image instance retrieval is the problem of retrieving images from a database representing the same object or scene as the one depicted in a query image. The first step of a typical retrieval pipeline starts with the comparison of vectors representing the image content known as *global image descriptors*. Deep neural networks have seen rapid progress in the last few years, and starting with their remarkable performance on the *ImageNet* large scale image classification challenge [1]–[3], they have become the dominant approach in a wide range of computer vision tasks. In recent work [4]–[7], Convolutional Neural Networks (CNN) have also been used to generate global feature descriptors for image instance retrieval, and are rapidly becoming the dominant approach for the retrieval problem.

While CNNs provide high performance, they suffer from one major drawback. State-of-the-art CNNs like AlexNet [1], VGG [2] and Residual Networks [3] consist of hundreds of millions of neurons. Stored in floating point precision, these networks require hundreds of megabytes for storage. Also, neural networks are getting deeper and deeper, as performance gains are obtained with increasing amounts of training data and increasing the number of layers: e.g., deep residual networks can be hundreds or even thousands of layers deep [3].

There are many practical reasons why smaller networks are desirable. First, there are several applications, where image classification or retrieval needs to be performed on a mobile device,

1. V. Chandrasekhar, L. Jie and Q. Liao contributed equally.

which require the CNN to be stored on the client. Mobile applications which are hundreds of megabytes in size are not practical. Second, as networks get larger, it is not feasible to train them on a single machine. Large neural networks are trained across multiple machines, and one of the key bottlenecks in training is the neural network weights or gradient data that are transferred between machines in the distributed gradient descent optimization step. Better weight compression will make training larger networks more practical. Third, there is immediate need for smaller networks for efficient hardware implementations of deep neural networks. Storing the entire network on chip will allow fast access, reduce processing latency and improve energy efficiency. Fourth, emerging MPEG standards like Compact Descriptors for Visual Search (CDVS) [8] and Compact Descriptors for Video Analysis (CDVA) require feature extraction to be performed with a few MB of memory to enable efficient implementations of streaming hardware. Without model compression, deep learning based descriptors cannot be adopted in these emerging standards.

Most of the recent work on model compression has been focused on compressing models for the image classification task: identifying the category that a query image belongs to. The two tasks: image classification and instance retrieval, while related, pose different requirements. CNNs consist of alternating convolutional and downsampling layers, and finally a set of one or more fully connected layers that map to a set of output classes or labels. For image classification tasks, the fully connected layers serve as the classifier, mapping rich feature representations to output classes. On the other hand, for image retrieval tasks, intermediate layers of the CNN have been shown to be effective high dimensional *global descriptors* [4].

We highlight some of the recent work on model compression, where the primary focus has been on reducing model size while maintaining high image classification accuracy. For architectures like [1], the fully connected layers contain the highest number of parameters. As a result, in [9], Gong et al. propose Vector Quantization techniques for parameters in the fully connected layers, while leaving the convolutional layers untouched. In their recent work, Han et al. [10], [11] propose quantization and coding techniques for compressing neural networks. Han et al. prune the network by learning only the important connections required for the classification task. Following pruning, scalar quantization and huffman coding are used to further reduce model size. In [12], the authors propose a new architecture called *SqueezeNet*, which contains $50\times$ fewer parameters than *AlexNet* [1], while achieving similar classification performance. The number of parameters in the network is reduced by smart choice of filter sizes and number of filters at each layer. 6-bit quantization is used to further reduce model size.

In this work, we propose quantization, pruning and coding techniques for compression of deep networks, with a focus on the image instance retrieval task, unlike [9], [10], [12]. We also study the problem in the context of state-of-the-art deep residual networks. For residual networks, we propose sharing parameters across different layers to reduce the number of weights. Quantization and coding techniques specific to residual networks are further applied to reduce model size. We perform extensive evaluation of trade-off between retrieval performance and model size for different types of networks on several data sets, providing the most comprehensive study on this topic. We compress models to the order of a few MBs: two orders of magnitude smaller than uncompressed deep networks, while achieving negligible loss in retrieval accuracy.

The paper is organized as follows. In Section 2, we discuss how CNNs are used for the image instance retrieval task. Following that, in section 3, we discuss quantization, coding, pruning and weight sharing techniques for state-of-the-art deep networks. In Section 4, we provide detailed experimental results and analysis of proposed methods.

2. Image Retrieval with Deep Networks

A typical image instance retrieval system starts with the comparison of vectors representing the image content, known as *global image descriptors*. There is a growing body of work focused

layer name	AlexNet	VGG-16	ResNet50	ResNet152	Shared ResNet
conv1	5×5, 96	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}$	7×7, 64	7×7, 64	7×7, 64
conv2_x	3×3, 256	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$
conv3_x	3×3, 384	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \\ 1\times1, 256 \end{bmatrix}$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times3$
conv4_x	3×3, 384	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \\ 1\times1, 512 \end{bmatrix}$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times10$
conv5_x	3×3, 256	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \\ 1\times1, 512 \end{bmatrix}$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$

Table 1. Architectures for different networks. Fully connected layers are discarded. Building blocks are shown in brackets, with the numbers of blocks stacked.

on using activations directly extracted from CNNs as *global descriptors* for image instance retrieval. All popular CNN architectures share a set of common building blocks: a succession of convolution-downsampling operations designed to model increasingly high-level visual representations of the data. Table 1 shows the model architecture for different networks considered in this work.

Initial studies [5], [13] proposed using representations extracted from fully connected layers of CNNs as a global descriptor for image retrieval. Promising results over traditional handcrafted descriptors like Fisher Vectors based on local SIFT features, were reported first in [4], [5], [13]. Recent papers [6], [14], [15] show that spatial max and average pooling of feature maps output by intermediate convolutional layers is an effective representation, and higher performance can be achieved compared to using fully connected layers.

Proposed techniques in [6], [14], [15] provide limited invariance to translation, but not to scale or rotation changes. To alleviate the scale issue, Tolias et al. [16] proposed averaging of max pooled features over a set of multi-scale region of interest (ROI) feature maps in the image, similar to the R-CNN approach [17] used for object detection. Inspired from a recently proposed invariance theory for information processing [18], we proposed a Nested Invariance Pooling (NIP) method to produce compact and performant descriptors, invariant to translation, scale and rotation [7].

In Figure 2, we provide a brief overview of NIP descriptors, which form the basis for all image retrieval experiments in this work. Figure 2(a) shows a single convolution-pooling operation for a single input layer and single output neuron. Figure 2(b) shows how a succession of convolution and pooling layers results in a set of feature maps f_i. A number of scale and rotation transformations are applied to the input image to obtain a series of feature maps. In Figure 2(c), we show how feature maps are pooled in a nested fashion by computing statistical moments at each step (average, standard deviation, max). The particular sequence of transformation groups and statistical moments are provided in [7], [19]. Key to achieving high performance is stacking multiple transformation groups in a nested fashion, and pooling with increasing orders of moments. Detailed evaluation provided in [19] shows that NIP is robust to scale and rotation changes, and significantly outperforms other CNN based descriptors. Next, we discuss compression of models shown in Table 1.

3. Model Compression

State-of-the-art CNNs commonly consist of alternating convolutional and downsampling layers, and finally one or more fully connected layers that map to a set of output classes or labels. Since

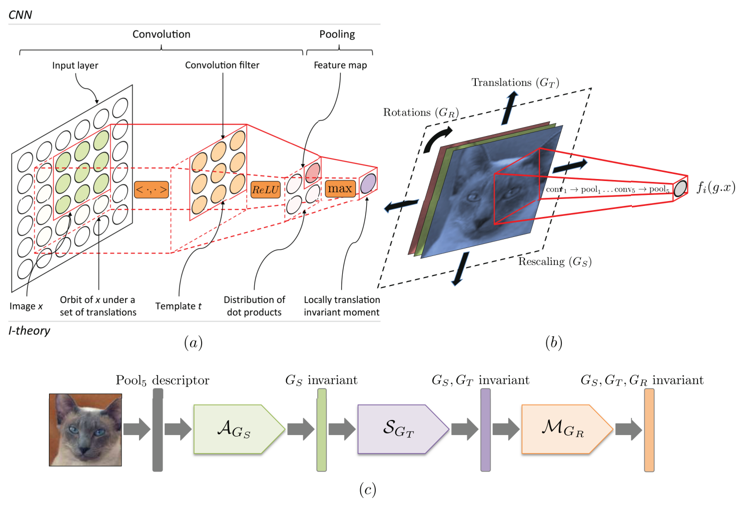


Figure 1. (a) A single convolution-pooling operation from a CNN schematized for a single input layer and single output neuron. (b) A specific succession of convolution and pooling operations learnt by the CNN (depicted in red) computes the feature, the last convolution layer, f_i for each feature map i from the RGB image data. A number of transformations g are applied to the input x in order to vary the response $f_i(g.x)$. (c) Starting with raw *pool5* descriptors, an arbitrary number of transformation group invariances are stacked up. G_S, G_T, G_R corresponds to transformation groups scale, translation and rotation respectively, while $A_{G_S}, S_{G_T}, M_{G_R}$ refers to statistical moments average, standard deviation and max respectively, which are computed in a nested fashion.

we are interested in the task of instance retrieval and not image classification, fully-connected layers which produce inferior global descriptors can be discarded. We consider the following class of networks in this work: *AlexNet* [1], *VGG* [2], 52-layer and 152-layer residual networks [3] (*ResNet*), and residual networks with shared parameters (*Shared ResNet*) [20]. Table 1 details the relevant part of the architecture of the different networks.

Table 2 lists the number of parameters in convolutional layers for each network. In Figure 2(a), we plot the number of parameters for each convolutional layer. We note there are millions of parameters in the convolution layers, and the number of parameters typically increases with depth. For *AlexNet* and *VGG*, the majority of parameters lie in the fully connected layers: discarding fully connected layers reduces the number of parameters in the network from 60M to 2.3M for *AlexNet* [1]. For residual networks, the number of parameters in the convolutional layers far exceeds the number of parameters in the fully connected layers, as there are many more convolutional layers and less fully connected layers. Even after discarding fully connected layers, uncompressed *VGG* and *ResNets* require more than 50 and 100 MB, motivating the need for compression. Next, we discuss four building blocks for model compression.

3.1. Quantization and Coding

In Figure 2(b) and (c), we show the distribution of weights in different convolutional layers *conv1* to *conv5* for *AlexNet*. We note that the convolutional weight parameters follow a Laplacian distribution. The distributions become more peaky (decreasing variance) with depth. This is intuitive as feature response maps become increasingly sparse with depth as higher level object representations are learnt. Similar trends are observed for other networks.

For each layer, we explore both scalar and vector quantization (VQ) techniques using the Lloyd-Max algorithm. For simplicity, we ignore entropy in the centroid training step. For VQ, we consider blocks of 2 and 4 with the number of codewords in the range of 256-1024. VQ

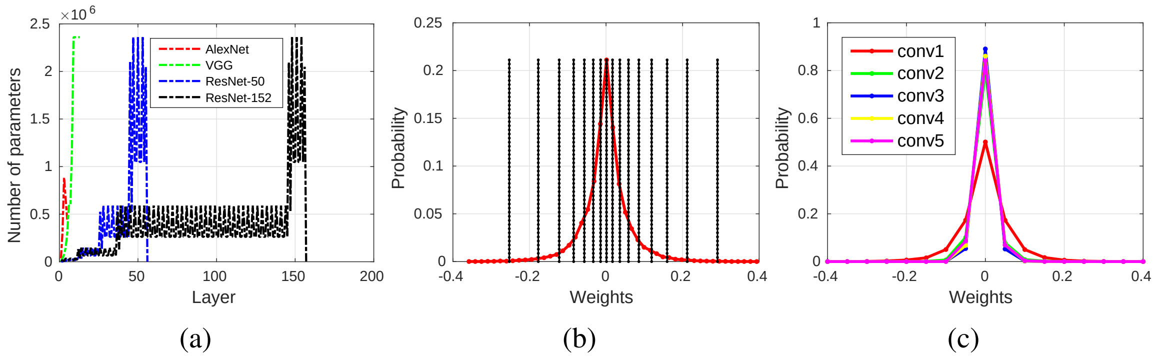


Figure 2. (a) Number of parameters in convolutional layers for different networks. (b) Distribution of weights for *conv1* layer of *AlexNet* and decision boundaries of Lloyd-Max quantizer. (c) Distribution of weights across different convolutional layers for *AlexNet*.

additionally requires the codebook to be transmitted along with the network. With increasing codebook size, the size of codebook data becomes non-negligble compared to the weights. In [9], Gong et al. focus on compressing fully connected layers with VQ with large codebooks, which is less applicable for compressing convolutional layers. Further, we explore applying different quantization parameters to different layers and study their impact on retrieval performance.

We also explore how variable length coding with Huffman codes can be used to further reduce model size. Variable length coding can reduce model size when models need to be transmitted over a network. However, it is not feasible to uncompress the network each time feature extraction needs to be performed, in which case, fixed length coding is preferred. Detailed experimental results are provided in Section 4.

3.2. Pruning

One technique for reducing model size is more coarse quantization of weight parameters. Another approach to trade-off model size for performance is to prune entire convolutional layers of the network, motivated by the visualization work in [21]. Zeiler and Fergus propose techniques for visualizing filter responses at different layers of a deep network [21]. The first convolutional layer typically learns Gabor-like filters, while the last layer represents high level concepts like cats and dogs. As we go deeper into the network, the representations become more specific to object categories, while earlier layers provide more general feature representations for instance retrieval. We reduce the number of parameters by varying the starting representation in Figure 2(c) for NIP based on earlier convolutional layers: *conv2* to *conv4*, instead of just the last layer before the fully connected layer: *pool5*. Layers after the chosen convolutional layer are dropped from the network. Note that this pruning approach is different from the pruning proposed in [10] where connections are removed if they do not impact classification accuracy.

3.3. Weight Sharing

We propose sharing weights across layers to reduce the number of parameters in residual deep networks (*ResNets*). A *ResNet* adopts repeated residual blocks to facilitate learning ultra deep representations. A residual block typically consists of two or more convolutional layers and a identity shortcut mapping connecting its beginning and end. A biological-inspired work [20] showed that repeated residual blocks is isomorphic to a specific type of recurrent network (RNN) unrolled over time (see Figure 4(a)). Enforcing weight sharing (like RNN does) across repeated residual blocks preserves the performance of the corresponding *ResNet* while significantly lowers the number of parameters. In our experiments, we build on [20] and repeat the residual blocks

	AlexNet	VGG	ResNet-52	ResNet-152	Shared ResNet
# of parameters	2.3M	14.7M	25.5M	60M	8.4M

Table 2. Number of parameters (only convolutional) for each model

conv1 to *conv4* 2, 3, 10 and 3 times respectively as illustrated in Figure 4(a). The resulting intermediate feature map sizes are also shown in Figure 4(a). These networks are trained on the ImageNet dataset, like the others. Quantization and coding are further applied to reduce model size.

4. Experimental Results

We study the trade-off between model size and performance on four popular instance retrieval data sets: *Holidays*, *Oxford buildings (Oxbuild)*, *UKBench (UKB)* and *Paris6k*. Following standard protocol for *Holidays*, *Oxford5k* and *Paris6k*, retrieval performance is measured by Mean Average Precision (mAP). For UKBench, we report the average number of true positives within the top 4 retrieved images (4×Recall@4). We start with off-the-shelf networks pre-trained on ImageNet classification data set. When comparing pooling layers to fully connected layers on *AlexNet*, we resize all images to (227×227) as fixed size input images are required. For all other experiments, if the longer side of input image exceeds 1024 pixels, we down sample the image to 1024, while maintaining aspect ratio.

To evaluate the performance of layer pruning, we choose 4 different layers for each network architecture. In particular, *pool1*, *pool2*, *conv3_relu*, *pool5* for *AlexNet*, *pool3*, *pool4*, *conv5a_relu*, *pool5* for *VGG*, *res3d_relu*, *res4c_relu*, *res4f_relu*, *pool5* for *ResNet-50* and *res3b7_relu*, *res4b15_relu*, *res4b35_relu*, *pool5* for *ResNet-152*. For *Shared ResNet*, we use the 4 layers following convolutional layers *conv1* to *conv4* shown in Table 1. For NIP feature extraction, we use 4 rotations (0 to 360 degrees with step size 90 degrees), and for each rotated image, we sample 20 ROIs with 3 different scales.

We study the trade-off between model size and performance in Figures 3, 4 and 5. For each curve, the model size is varied by pruning each network back to intermediate layers as discussed above. Different curves represent compression with different quantization parameters. We make several interesting observations.

- We first compare the performance of the last convolutional layer *pool5* and full connected layers *fc6*, *fc7* and *fc8* for *AlexNet* on the *Holidays* data set in Figure 3(a). Performance for *pool5* is higher than the fully connected layers. A significant drop is observed for *fc8*, which represents the layer corresponding to *ImageNet* class labels. This confirms that fully connected layers can be discarded for the instance retrieval problem, while drastically reducing network size. Similar trends are observed on other data sets, and confirms observations made in [15].
- We compare Scalar Quantization (SQ) and Vector Quantization (VQ) for *AlexNet* on the *Holidays* data set in Figure 3(b). *k64-b2* for VQ corresponds to codebook size of 64 and block size of 2. We note that there is only a small gain with VQ compared to 4-bit or 5-bit SQ. This is intuitive, as one can expect weight parameters to be independent, and large codebooks for VQ are not feasible. From here on, we use SQ in the rest of the experiments.
- We compare Fixed Length Coding (FLC) and Variable Length Coding (VLC) for *AlexNet* on the *Holidays* data set in Figure 3(c). We observe a small but consistent gain of ∼15-20% with variable length coding, which can be useful when models need to be transmitted over a network. From here on, we use FLC in the rest of the experiments.
- For compression of residual networks, we make two key observations. First, we observe that

Layer	Resnet-50	ResNet-50 4-bit Conv 32-bit BN	ResNet 4-bit Conv 4-bit BN
pool5	0.8382	0.7806	0.7591

Table 3. Coarse quantization of Batch Normalization (BN) parameters leads to drop in performance.

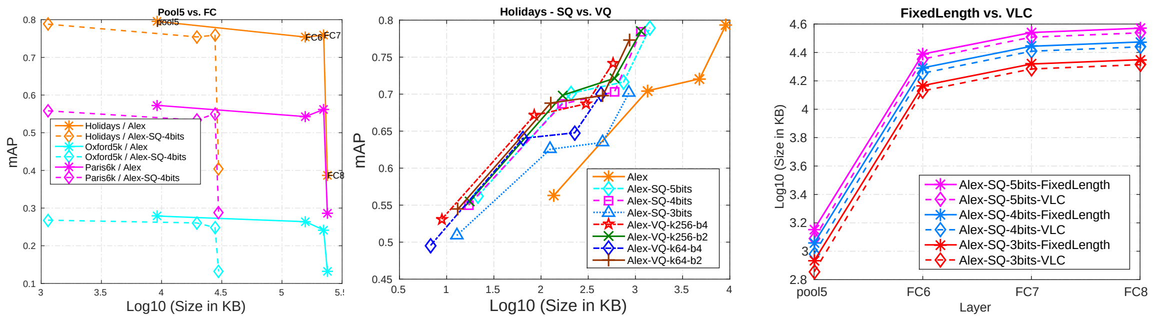
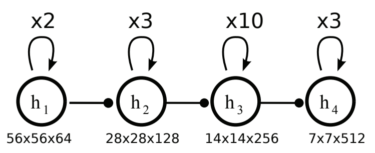


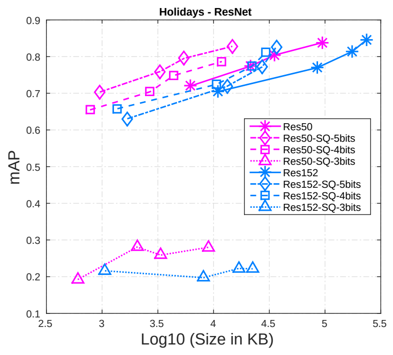
Figure 3. (a) Comparing *pool5* and fully connected layers on *AlexNet*. (b) Comparing SQ and VQ for varying parameters on *AlexNet*. (c) Comparing fixed length and variable length coding on *AlexNet*.

performance drops drastically for 3-bit quantization for residual networks in Figure 4(b), compared to *AlexNet* in Figure 3(b)-(c). Residual networks are a lot deeper, and the quantization error accumulates over the layers leading to the steep drop. The residual networks in consideration also have batch normalization layers, which zero-center the data at every layer. Quantizing the Batch Normalization (BN) weights coarsely results in a drop in performance as shown in Table 3. As a result, we maintain the BN weights with floating point precision, while coarsely quantizing weight parameters in the rest of the experiments.

We compare different model compression schemes in Figure 5 and make the following salient observations.

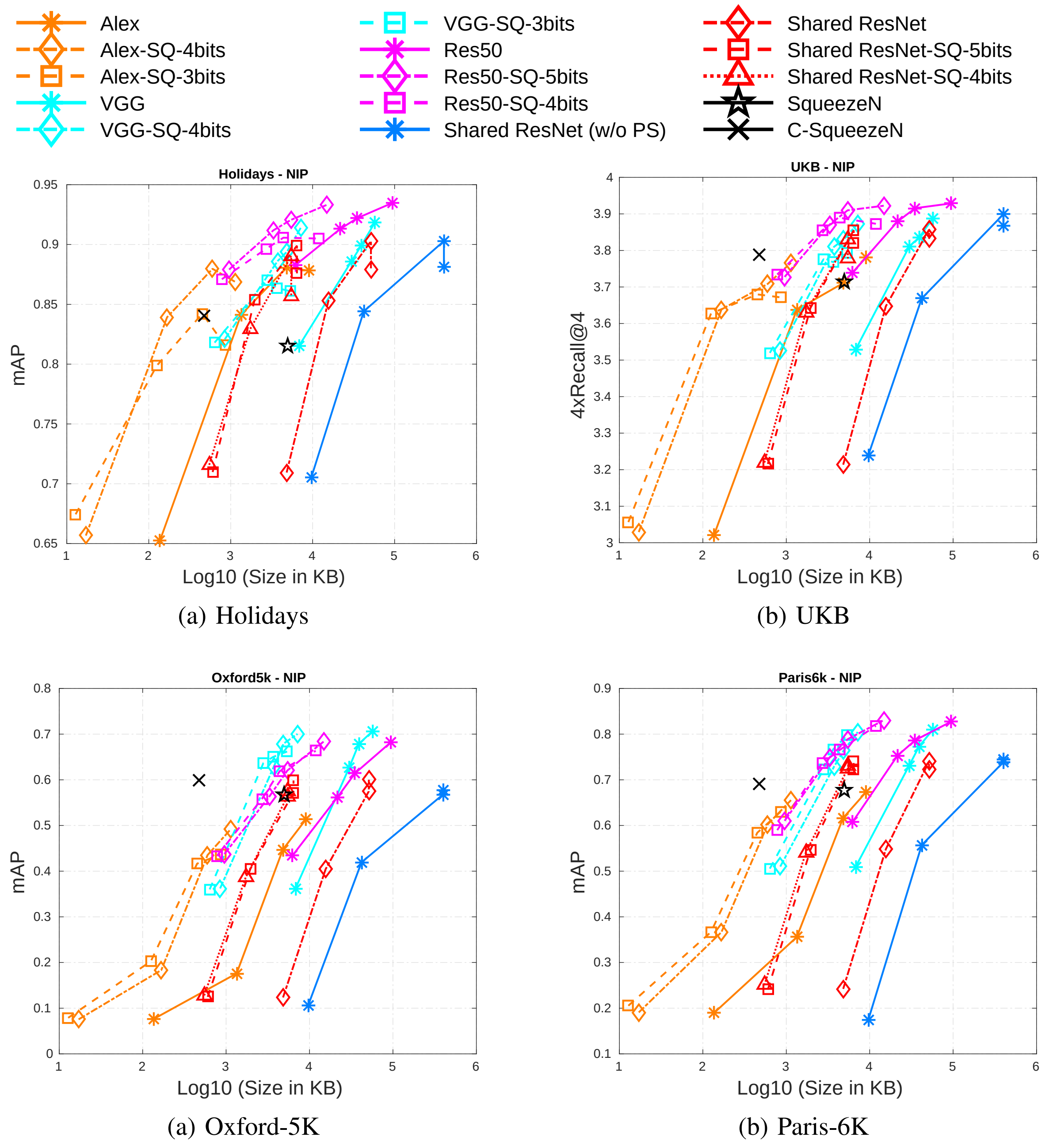
- Points 3,4,5 on the x axis correspond to 1, 10 and 100 MB models respectively. We observe that there is negligible loss in retrieval performance with 4-bit quantization for *AlexNet* and *VGG*. For residual networks, this is achieved with 5-bit quantization. 2-bit quantization results





(a) (b)

Figure 4. (a) Weight sharing across layers in residual networks. Building blocks in Table 1 *SharedNet* are repeated 2, 3, 10 and 3 times respectively with shared weights. (b) Effect of scalar quantization on residual networks. Retrieval performance drops steeply for 3-bit quantization.



(a) Holidays (b) UKB

(a) Oxford-5K (b) Paris-6K

Figure 5. Comparing different model compression schemes on 4 different data sets. Points 3, 4 and 5 on the x axis correspond to 1 MB, 10 MB and 100 MB respectively. For each curve, model size is varied by pruning layers in the network.

in significant drop in performance for all models, and are not shown in the interest of space. Compressed models in the order of 5-10 MB achieve performance almost similar to their uncompressed counterparts on each data set while being one to two orders of magnitude smaller.

- Both pruning layers and coarser quantization of weight parameters provide similar trade-off in performance and model size.
- Comparing compressed models, we note that compressed *VGG* achieves the highest performance on *Oxford5k*, while compressed *ResNet* achieves the highest performance on the other three data sets. Compressed *AlexNet* results in smaller networks, but performs significantly worse than top performing schemes.
- Compressed *SqueezeNet* [12] (C-SqueezeN) models are only 0.5 MB in size, and provide good trade-off in size and performance. However, they perform significantly worse than the best schemes. The aggressive optimizations used for reducing parameters in the *SqueezeNet* architecture hurt retrieval performance significantly.
- For *Shared ResNet*, we first compare the models with and without parameter sharing, denoted as *Shared ResNet* and *Shared ResNet w/o PS* respectively. Compared to no parameter sharing, 3-4 $\times$ smaller models are achieved by sharing weights. Further quantization of *Shared ResNet* results in smaller models. One thing to note is that the *Shared ResNet* are trained from scratch on the *ImageNet* data set, and peak performance for our trained models is lower than the uncompressed *ResNet-50* of [3]. Performance of compressed *Shared ResNet* can be further improved by starting with better residual network models. An alternative approach is to enforce shared weights across layers in the *ResNet* models of [3] and fine-tune the network again.

The experimental results above provide the most comprehensive study of model compression for the instance retrieval problem. We provide several interesting directions for future work. Models need to be made smaller further without losing retrieval performance: e.g., the Compact Descriptors for Visual Search (CDVS) standard only allows for 1 MB of memory for feature extraction in the low memory mode to enable streaming hardware implementations. Mathematical modeling of the distribution of weights would provide interesting insights for learning more compact models. More sophisticated pruning techniques can be explored to reduce model size, with the explicit objective of maintaining high retrieval performance. Performance of deep residual networks with shared weights needs to be improved further, and is a promising direction for achieving smaller performant models.

5. Conclusion

In this work, we studied the problem of neural network model compression focusing primarily on the image instance retrieval task. We studied quantization, coding, pruning and weight sharing techniques for reducing model size for the instance retrieval problem. Our compressed models are in the order of a few MBs: two orders of magnitude smaller than the uncompressed models while achieving negligible loss in retrieval accuracy. We provide extensive experimental results on the trade-off between retrieval performance and model size for different types of networks on several data sets, providing the most comprehensive study on this topic.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[2] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in *arXiv preprint arXiv:1409.1556*, 2014, pp. 1–10. [Online]. Available: http://arxiv.org/abs/1409.1556

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[4] V. Chandrasekhar, L. Jie, O. Morere, H. Goh, and A. Veillard, “A Practical Guide to CNNs and Fisher Vectors for Image Instance Retrieval,” in *Signal Processing, Elsevier*, 2016.

[5] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, “Neural Codes for Image Retrieval,” in *Proceedings of European Conference on Computer Vision (ECCV)*, 2014.

[6] A. Sharif Razavian, J. Sullivan, A. Maki, and S. Carlsson, “A baseline for visual instance retrieval with deep convolutional networks,” in *International Conference on Learning Representations, May 7-9, 2015, San Diego, CA*. ICLR, 2015.

[7] O. Morère, A. Veillard, J. Lin, J. Petta, V. Chandrasekhar, and T. A. Poggio, “Group invariant deep representations for image instance retrieval,” *CoRR*, vol. abs/1601.02093, 2016. [Online]. Available: http://arxiv.org/abs/1601.02093

[8] L. Duan, V. Chandrasekhar, J. Chen, L. Jie, Z. Wang, T. Huang, B. Girod, and W. Gao, “Overview of the mpeg cdvs standard,” *IEEE Transactions on Image Processing*, 2015.

[9] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev, “Compressing deep convolutional networks using vector quantization,” *CoRR*, vol. abs/1412.6115, 2014. [Online]. Available: http://arxiv.org/abs/1412.6115

[10] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *International Conference on Learning Representations (ICLR)*, 2016.

[11] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 1135–1143.

[12] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.

[13] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition,” in *Computer Vision and Pattern Recognition*, Mar. 2014. [Online]. Available: http://arxiv.org/abs/1403.6382 http://adsabs.harvard.edu/abs/2014arXiv1403.6382S

[14] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, “From generic to specific deep representations for visual recognition,” in *Computer Vision and Pattern Recognition Workshops*, 2015.

[15] A. Babenko and V. Lempitsky, “Aggregating local deep features for image retrieval,” in *International Conference on Computer Vision (ICCV)*, 2015.

[16] G. Tolias, R. Sicre, and H. Jégou, “Particular object retrieval with integral max-pooling of CNN activations,” in *arXiv:1511.05879*, 2015.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[18] F. Anselmi, J. Z. Leibo, L. Rosasco, J. Mutch, A. Tacchetti, and T. Poggio, “Unsupervised learning of invariant representations in hierarchical architectures,” *arXiv:1311.4158*, 2013.

[19] O. Morère, J. Lin, A. Veillard, and V. Chandrasekhar, “Nested invariance pooling and RBM hashing for image instance retrieval,” *CoRR*, vol. abs/1603.04595, 2016. [Online]. Available: http://arxiv.org/abs/1603.04595

[20] Q. Liao and T. A. Poggio, “Bridging the gaps between residual learning, recurrent neural networks and visual cortex,” *CoRR*, vol. abs/1604.03640, 2016. [Online]. Available: http://arxiv.org/abs/1604.03640

[21] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” *CoRR*, vol. abs/1311.2901, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901