OMCBIR: Offline mobile content-based image retrieval with lightweight CNN optimization

X. Zhang, C. Bai, K. Kpalma

OMCBIR: Offline mobile content-based image retrieval with lightweight CNN optimization

Xiaoqing Zhang[a], Cong Bai[a,*], Kidiyo Kpalma[b]

[a]College of Computer Science and Technology, Zhejiang University of Technology,Hangzhou 310023, China
[b]INSA Rennes, CNRS, IETR - UMR 6164, Univ Rennes, Rennes F-35000, France

Communicated by S. Sarkar

Keywords:
Image retrieval
Model compression
Lightweight neural network
Mobile devices

A B S T R A C T

Convolutional Neural Networks (CNNs) have achieved great success in computer vision applications. However, due to the high requirements for computation power and memory usage, most state-of-the-art CNNs are difficult to deploy on resource-constrained mobile devices. Although many typical lightweight neural networks have been proposed in the industry, such as MobileNetV2, which reduce the amount of parameters and calculations, they still have a lot of redundancy. Furthermore, few papers consider the use of deep learning models to implement image retrieval on terminals, so we propose a new offline retrieval framework based on lightweight neural network models, called Offline Mobile Content-Based Image Retrieval (OMCBIR). In this framework, we focus on the feature extraction of the model, by introducing pointwise group convolution and channel shuffle into the bottleneck block, reconstructing the network structure, and introducing the convolutional attention module, we propose an extremely lightweight small network-Attention-based Lightweight Network(ALNet). Compared to MobileNetV2, ALNet obtains a higher mAP on each dataset in OMCBIR when the model parameters are reduced by more than 62% and the model size is reduced by more than 63%. Extensive experiments conducted on five public datasets provide a trade-off between retrieval performance and model size of different algorithms, which proves the efficiency of the proposed OMCBIR.

1. Introduction

Content-based image retrieval (CBIR) aims to retrieve quickly and accurately the images needed by the users from a large number of digital images by extracting the visual information of the image, which is of great value in both research and commercial use. In the past ten years, with the advent of deep learning, the features learned from convolutional neural networks (CNNs) have been widely applied to various visual tasks. Babenko *et al*. [1] is the first attempt to fine-tune a CNN model for image retrieval. Currently, CNN features have been used as a general representation of CBIR.

However, most CNNs have a large amount of calculation and parameters, which can only be used on the server side and require high-performance accelerator to run. Thus there is a serious contradiction between complex models and limited computing resources. Due to the limitation of storage space and power consumption, storage and calculation of deep neural network models on embedded devices and mobile terminals are still facing huge challenges. At the same time, the era of the Internet of Things has come, and the number of mobile smart applications has exploded. Internet of Things applications have put forward higher requirements for response time, privacy data protection, and big data transmission. The efficiency of cloud computing is not enough to support these applications, and the edge of the network is changing from data consumer to data producer as well as data consumer [2]. Therefore, how to ap-

*Corresponding author:
e-mail: congbai@zjut.edu.cn (Cong Bai)

ply a lightweight CNN on a specific task to the terminals has become an urgent problem. At present, most of the existing image retrieval methods are aimed at online retrieval problems, and offline mobile image retrieval (OMIR) based on deep learning models is still not fully explored.

In this paper, we intend to fill this gap and focus on OMIR methods based on lightweight neural network. Compared with online image retrieval, OMIR is not restricted by the network, guarantees the privacy and security of data, and can complete the retrieval in a very short time. It implements image retrieval by applying deep learning models to terminal devices in an offline manner. OMIR needs to solve two main problems: one is to build a retrieval framework that can be used to implement offline retrieval, and the other is to train a deep learning model to obtain image features. After that, image retrieval is completed by converting the deep learning model into a terminal model and embedding it into the retrieval framework.

To solve the above problems, we propose a novel offline retrieval framework based on a lightweight neural network, called OMCBIR, as shown in Figure 1. OMCBIR builds a lightweight convolutional neural network as a feature extractor to obtain image features, which is the core part of the framework. With the help of OMCBIR, we can implement offline image retrieval on terminal devices.

However, the most important factor affecting retrieval performance is feature extraction, so we focus on improving this module. Although some lightweight CNNs have been proposed, they still suffer from a large amount of expensive computing power and memory usage. To further reduce the model parameters and computational complexity, we propose an Attention-based Lightweight Block (ALBlock) unit. ALBlock significantly reduces the amount of computation and parameters within the block by grouping a large number of 1×1 convolutions in the bottleneck block [3]. At the same time, channel shuffle is used to make up for the information flow between channel groups and improve the feature representation ability. Furthermore, based on the proposed ALBlock unit, this paper proposes an extremely lightweight network architecture, called Attention-based Lightweight Network (ALNet), which combines an attention mechanism to improve retrieval performance. Finally, the model demonstrates good retrieval performance in OMCBIR with greatly reduced model parameters and computation cost. In summary, the main contributions of this paper are as follows.

1) We propose a novel offline framework for content-based image retrieval on the mobile side, namely OMCBIR. As far as we know, this is the first time that a deep learning model is used for a completely offline mobile content-based image retrieval task.

2) As the core of OMCBIR, we propose an extremely lightweight network architecture ALNet, whose role is to extract features from images. Besides, we also propose an ALBlock unit, which is the basic building block of ALNet. By introducing pointwise group convolution, channel shuffle and convolution attention module in bottleneck block, we not only improve retrieval accuracy, but also greatly reduce the computational cost and memory usage of the model.

3) The experimental results on five public datasets demonstrate the effectiveness of our model. Compared to MobileNetV2, ALNet obtains a higher mAP on each dataset in OMCBIR with over 62% reduction in model parameters and over 63% reduction in model size.

2. Related work

2.1. Deep Learning based Online Image Retrieval

In the last decade, following the emergence of deep learning, feature representation begin to shift from hand-crafted to learning-based [4], in which feature learning based on CNNs replaces traditional hand-crafted feature representation as the most advanced pipeline. Deep learning is a hierarchical feature representation technology used to learn abstract features that are important to datasets and applications from data. At present, CNN features have been used as a general expression for content-based image retrieval, and a lot of researches are devoted to obtain image features that are conducive to retrieval for online image retrieval. For example, previous work [5] optimizes AlexNet from the pooling layer, fully connected layer, and hidden layer, and obtains state of the art results on large-scale image retrieval tasks.

The fixed-length compact feature vector is usually generated by other measures of pooling or convolutional layer. CNN-based image representation taking advantage of deep learning, can be regarded as a global feature, and has achieved remarkable performance in image retrieval. [6] points out that many methods directly use pool strategy to obtain image features and successfully perform image search, such as global maximum pool, global average pool, CroW pool [7], R-MAC pool [8], etc. For example, Razavian *et al*. [9] segments the original image into many blocks, and then extracts features from the segmented image blocks. Finally, the feature vectors extracted from all the blocks of the image are put together for post-processing. Babenko *et al*. [10] aggregate local depth features to generate compact global descriptors for image retrieval. Kalantis *et al*. [7] mainly propose a direct and effective image representation method based on the cross-dimensional weighting and aggregation of the output of the deep convolutional neural network layer. Tolias *et al*. [8] construct a compact feature vector to encode multiple image regions without providing multiple inputs to the neural network, and proposes an R-MAC pool method for image retrieval.

With the popularity of Generative Adversarial Networks [11, 12] (GANs), many researchers have explored the possibility of using GAN in image-related applications because it shows great potential. Some of the existing research methods tend to use GAN to obtain more training data. By using GAN to generate synthetic images that are visually similar to the input image, this is conducive to image retrieval tasks, because the ultimate goal is to retrieve similar images from a given dataset. On the other hand, a lot of research is devoted to use GAN to realize unsupervised image retrieval. E.g, Dizaji *et al*. [13] propose HashGAN, which can effectively obtain image binary representation without pre-training. In addition, a novel hash loss function and cooperative loss function are introduced to achieve

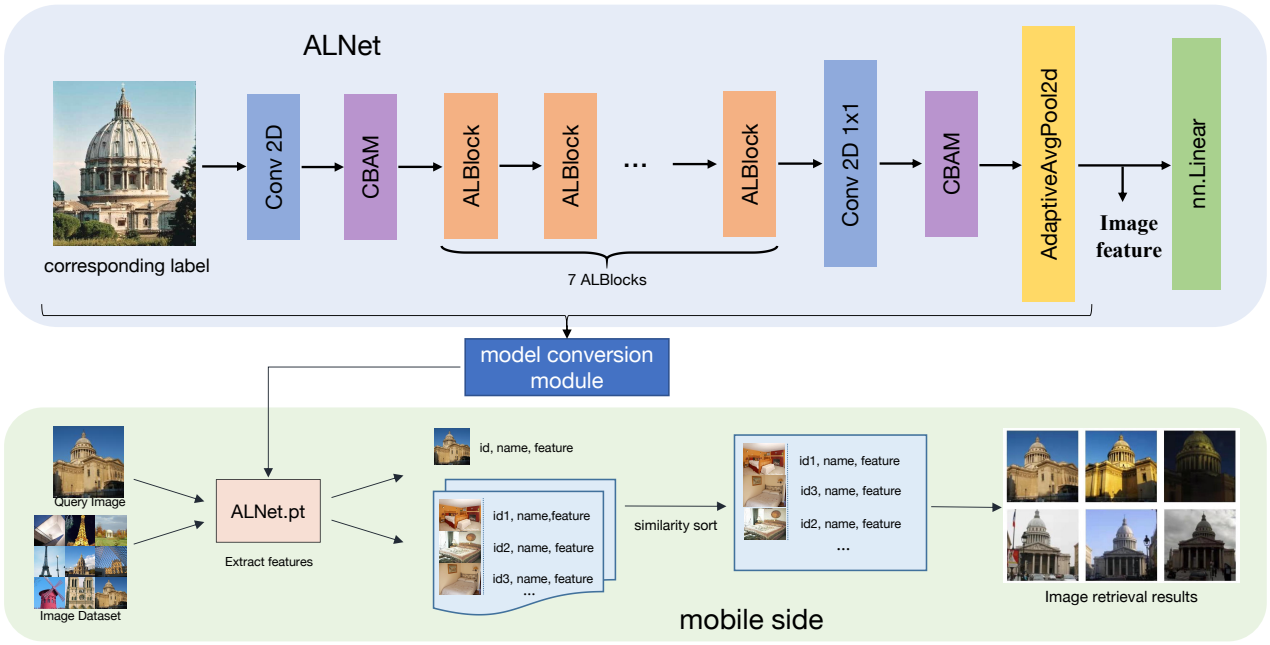

Fig. 1. The framework of the proposed OMCBIR. Model training stage: The parameters of the convolutional layer trained on each dataset are saved as a pre-training model. Offline retrieval stage: The pre-training model can be run on the mobile terminal through model conversion technology, and the retrieval result can be obtained by measuring the L2 distance between the query image and the image in the vectorized dataset.

similar random input and hash bits of the synthesized image. Song *et al*. [14] propose a binary generative confrontation network BGAN+, which converts images into binary codes, and performs image retrieval and compression in a multi-task learning manner. Bai *et al*. [15] propose an unsupervised framework for adversarial instance-level image retrieval, which is the first time that adversarial training is adopted in retrieval procedure at instance level image retrieval task.

CNNs have reached the most advanced accuracy in many computer vision tasks. However, the success of CNN usually depends on a large amount of computation and memory consumption, which limits the use of CNNs on resource-limited devices such as mobile or embedded devices. In order to solve these problems, efforts are also made in the field of deep learning to promote the miniaturization of neural networks.

2.2. Deep learning based Mobile Image Retrieval

Currently, mobile image retrieval based on deep learning can be divided into online and offline methods. The similarity between the two is that a pre-trained DNN model is used to extract feature vectors of terminal images. The difference is that online mobile image retrieval transmits the extracted features to the server, conducts image retrieval on the cloud, and finally returns the retrieval results to the terminal for display. This semi-offline approach requires frequent interaction with the server. However, our proposed OMCBIR can complete absolute offline retrieval, which is not limited by the network, can ensure the privacy and security of data, and reduce the inference time delay caused by multiple interactions.

When using deep learning on mobile, it is often necessary to compress CNNs [16], including network pruning [17, 18, 19, 20], parameter quantization [17, 21], knowledge distillation [22, 23, 24, 25] and lightweight network design [26, 27, 28, 29, 30, 3, 31, 32]. Network pruning uses a specific algorithm to cut out redundant network parameters without affecting the performance of the neural network. The common steps of neuron pruning include training, pruning and fine-tuning, in which the latter two steps are usually repeated many times. Parameter quantization is a method of compressing the network by reducing the number of bits required to represent each weight, including low-bit quantization, binary quantization and etc., which can greatly improve efficiency and make it possible to deploy deep learning algorithms in actual scenarios. However, low-precision weight representation may cause performance degradation [33]. Hinton *et al*. [22] put forward the concept of knowledge distillation for the first time, and its main design idea is to use the trained teacher model to guide the training of the student model, thus achieving knowledge transfer. Knowledge distillation is suitable for the training of small-volume models, but the artificial setting of student network results is very subjective, which will make the training effect very different.

Compared with three methods mentioned above, designing a lightweight network is the most direct and effective way to reduce the size of the model, which is directly applicable to mobile and embedded devices. The main idea of lightweight model design is to design a more efficient network calculation method, so that network parameters are reduced without loss of network performance. Popular lightweight neural network models are SqueezeNet [26], Xception [27], ShuffleNet series [28, 29], EfficientNet [32], MobileNet series [30, 3, 31]. The advantage of SqueezeNet lies in the design of the squeeze layer and the expand layer, which reduces the amount of parameters by replacing the 3×3 convolution with 1×1 convolution and

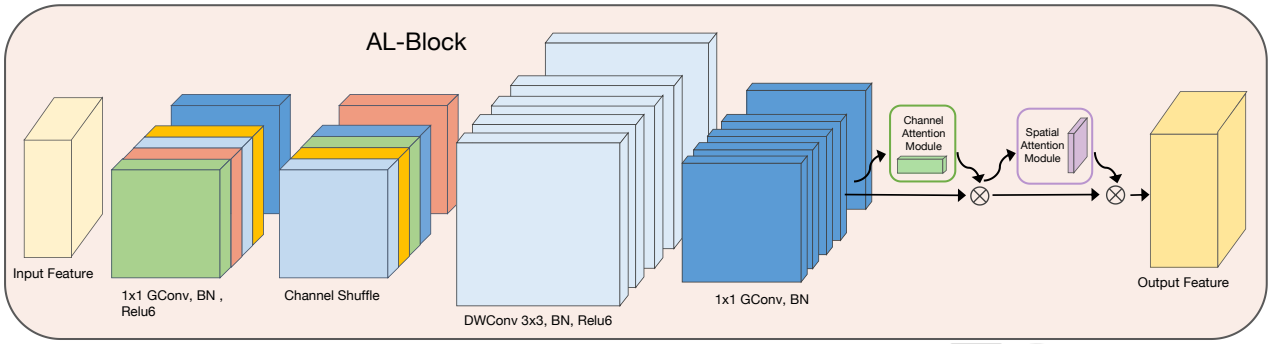

Fig. 2. Attention-based lightweight block(ALBlock). The first four parts in the block perform convolution operations in sequence. Finally, use the output feature map as the input of the convolutional attention module to get the final output.

reducing the number of input channels in the expand layer. The basic idea of Xception is to design a channel-separated convolution. Because there are a large number of dense 1×1 convolutions in Xception, the network is very inefficient. Therefore, ShuffleNet network that combines point-by-point group convolution and channel shuffle is proposed. Based on neural structure search technology, EfficientNet finds a simple and efficient composite coefficient to enlarge the network from three dimensions of depth, width and resolution to improve the accuracy of the model, which requires a lot of GPUs. MobileNet uses depth-wise separable convolution to build a lightweight deep neural network. On the basis of depth-wise separable convolution, MobileNetV2 introduces two modules: inverted residual and linear bottleneck, which makes the model smaller and faster when the network is deeper, but it still has a large amount of expensive point-by-point convolutions. Therefore, the parameters and computational complexity of MobileNetV2 need to be further reduced in order to better apply it to mobile devices with limited memory.

3. Methodology

We propose an offline mobile content-based image retrieval (OMCBIR) framework based on a lightweight neural network. In this framework, the most important factor affecting deployment and retrieval is the feature extraction module. In order to further compress the model and to obtain image features that are conducive to retrieval, we focus on the improvement of feature extraction module, and propose ALBlock unit and ALNet lightweight neural network to further reduce the computation and parameters of the model. The core details of this framework will be elaborated in the following sections.

3.1. Framework Overview

The overall framework proposed in this paper is shown in Figure 1. The whole OMCBIR consists of two important stages. The first stage is to design and to train a more lightweight network, with the purpose of extracting features. The second stage is image retrieval, including model conversion and deployment. After that, the loaded model is used to extract features in batches to complete image retrieval. The entire framework uses this model for classification training, saving the model parameters with the highest classification accuracy. Before migrating to mobile, we load the pre-trained model, remove the classifier, and perform a model conversion so that the model can be used for image retrieval on mobile device.

3.2. Efficient Lightweight Block Design

We find that the bottleneck structure of MobileNetV2 still has a lot of expensive point-by-point convolutions, and the amount of calculation is still not negligible. In order to further compress the model, we propose an improvement to the bottleneck, named ALBlock. The overview of ALBlock is shown in Figure 2. Firstly, we use group convolution (GConv) to effectively reduce the computation and parameter amount of point-by-point convolution. However, grouping will lead to the failure to exchange the feature information of different groups, which will weaken the performance of the model to a certain extent. In order to solve this problem, we introduce channel shuffle after the first grouped convolutional layer. Channel shuffle reorganizes the feature map after group convolution, which can improve the expression ability of features and model generalization without increasing the amount of calculation by performing uniform channel exchange in different groups. The third layer of ALBlock is a depthwise convolutional layer. Unlike conventional convolution operations, a convolution kernel of depthwise Convolution (DWConv) is only responsible for one channel. The depthwise convolution is followed by a second grouped convolutional layer, which performs a weighted combination of the feature maps from the previous step in the depthwise direction.

In order to improve the feature learning ability of the model within the block, we also introduce a lightweight Convolutional Block Attention Module(CBAM) after the second grouped convolutional layer. Therefore, we not only reduce the amount of parameters and computation, but also preserve the accuracy of the model [34].

For the Bottleneck Block of MobileNetV2 and ALBlock, we suppose the size of the input feature map is $H_1 \times W_1 \times C_1$, the size of the output feature map is $H_2 \times W_2 \times C_2$, the number of input and output channels of the internal Dwise is tC_1, where H_i, W_i, and C_i ($i = 1$ or 2) represent respectively height, width and number of channels of the feature map, and t represents the expansion multiple of the Dwise convolution. The first layer

Table 1. Calculation of Parameter Analysis of Bottleneck and ALBlock.

	Each layer	FLOPs	Params
Bottleneck Block of MobileNetV2	1×1 Conv	$C_1 \times H_1 \times W_1 \times tC_1$	$C_1 \times tC_1$
	DWConv 3×3	$3 \times 3 \times tC_1 \times H_2 \times W_2$	$3 \times 3 \times tC_1$
	1×1 Conv	$tC_1 \times H_2 \times W_2 \times C_2$	$tC_1 \times C_2$
	Sum	$tC_1(C_1 \times H_1 \times W_1 + (C_2 + 9) \times H_2 \times W_2)$	$tC_1(C_1 + C_2 + 9)$
ALBlock	1×1 GConv	$\frac{1}{g} \times C_1 \times H_1 \times \frac{1}{g} \times W_1 \times tC_1 \times g$	$\frac{1}{g} \times C_1 \times \frac{1}{g} \times tC_1 \times g$
	DWConv 3×3	$3 \times 3 \times tC_1 \times H_2 \times W_2$	$3 \times 3 \times tC_1$
	1×1 GConv	$\frac{1}{g} \times tC_1 \times H_2 \times \frac{1}{g} \times W_2 \times C_2 \times g$	$\frac{1}{g} \times tC_1 \times \frac{1}{g} \times C_2 \times g$
	Sum	$tC_1[\frac{1}{g} \times C_1 \times H_1 \times W_1 + (\frac{1}{g} \times C_2 + 9) \times H_2 \times W_2]$	$tC_1[\frac{1}{g} \times (C_1 + C_2) + 9]$

of the Bottleneck of MobileNetV2 is the standard convolution operation, where the convolution kernel is 1×1, the number of input channels is C_1 and the number of output channels is tC_1. The second layer is DWConv, where a convolution core of depth convolution is responsible for a channel, the convolution core is 3×3, and the number of input and output channels are tC_1. The third layer is still a standard convolution operation. The size of the convolution kernel is 1×1, the number of output channels is C_2.

After improvement, ELBlock divides the extended 1×1 Conv into g groups, so the size of each input feature map in the first layer is $H_1 \times W_1 \times C_1/g$, and the corresponding convolution kernel size is C_1/g.The computational complexity of the channel shuffle is very low, so it is not considered. The DWConv layer of the second layer is the same as the bottleneck structure in MobileNetV2. and the convolution kernel of 1×1GConv in the third layer is C_2/g. The calculation amount and parameter amount of each layer of Bottleneck of MobileNetV2 and ALBlock and the sum of that are shown in Table 1:

It can be seen that the ratio of the parameters of the improved ALBlock to the original Bottleneck is:

$$R_1(g) = \frac{\frac{1}{g} \times (C_1 + C_2) + 9}{C_1 + C_2 + 9} \tag{1}$$

The ratio of the FLOPs is:

$$R_2(g) = \frac{\frac{1}{g} \times (C_1 \times H_1 \times W_1) + (9 + \frac{C_2}{g}) \times H_2 \times W_2}{C_1 \times H_1 \times W_1 + (9 + C_2) \times H_2 \times W_2} \tag{2}$$

It can be seen from formulas (1) and (2) that when the number of groups is g=8 that is used in the proposal, the parameter ratio R_1 is:

$$\frac{\frac{1}{8} \times (C_1 + C_2) + 9}{C_1 + C_2 + 9} \tag{3}$$

and the FLOPs ratio R_2 is:

$$\frac{\frac{1}{8} \times (C_1 \times H_1 \times W_1 + (9 + \frac{C_2}{8}) \times H_2 \times W_2}{C_1 \times H_1 \times W_1 + (9 + C_2) \times H_2 \times W_2} \tag{4}$$

and both have achieved a significant drop.

3.3. Structure of ALNet

This subsection describes the structure of ALNet in detail. As shown in Figure 1, the input of ALNet is the image and the corresponding label, and the output is the probability that the image belongs to a certain class. We improve and optimize the framework from two aspects: block internal optimization and network structure. The first layer of ALNet uses standard 3×3 convolutions to construct the initial filter, which plays the role of edge detection [31]. CBAM is added after the first layer to improve the learning of key features, the detailed description of CBAM is given in subsection 3.4. Then followed by seven improved ALBlocks, which are stacked in sequence. ALBlock layer is followed by 1×1 convolution layer, which is equivalent to the full connection layer, realizing the dimension raising of the feature channel, increasing the nonlinearity of the increasing network, and enabling the network to express more complex features. We add another CBAM module after the 1×1 convolutional layer, and finally, replace the average pooling layer with a binary adaptive average pooling layer (AdaptiveAvgPool2d) to obtain image features of the same size for image retrieval. When the classification training is over, the desired model features is extracted by removing the classifier of the pre-trained model instances.

The network structure of ALNet adopts a single stack mode, and the basic building block is ALBlock, so as to achieve the effect of reducing the amount of model calculation without losing the retrieval accuracy.

3.4. Attention Mechanism

In order to improve the performance of deep neural networks, many researches focus on increasing the network depth, width and cardinality, but these need to occupy more resources. Recently, Woo *et al.* [34] propose a plug-and-play lightweight CBAM, which is mainly characterized by an attention mechanism. CBAM uses the original convolution operation to extract features by mixing cross-channel and spatial information, and can enhance meaningful features on the channel and spatial axis dimensions by sequentially applying channel and spatial attention modules with only a small computational cost. Attention not only tells the neural network model where to focus, it also improves the representation of interests.

CBAM can extract more detailed features and improve the representation ability of convolutional networks. And CBIR always relies on the ability of descriptors to represent images, so this paper inserts the CBAM module into the model to enhance the feature expression of the model. Combined with the model

mentioned in this paper, we not only add CBAM to the model's architecture, but also add it to each ALBlock.

3.5. Implementation Details

In the proposed OMCBIR, the model parameters of ALNet will be saved with the highest classification accuracy during the entire training process. When the training epoch reaches the preset value, we terminate the training. Before the model conversion, the classifier of the model is removed with the aim of obtaining the output features by forward inference of the model. After that, the model conversion provided by the Pytorch Mobile framework is used to complete the conversion of the PyTorch model to the TorchScript model, which supports the efficient operation of machine learning models on edge devices. Finally, we deploy the well trained model to OMCBIR, by using PyTorch Mobile's java interface to call the model in the Android project.

In OMCBIR, we extract the image features of the image database in batches and store them in the local database. When performing image retrieval, we use the loaded deep learning model to extract the features of the query image, compare them with the features in the local database, obtain pictures similar to the query image according to the similarity matching algorithm, and calculate mAP. In algorithm 1, we summarize the workflow of OMCBIR.

Algorithm 1 OMCBIR's workflow.

Input: training data X and training epoch $epoch_{max}$.
Output: The compact model and its parameters W^*.

```
Initialize the weights of model;
for epoch = 1 → epoch_max do
    Update the model parameter W based on X;
    Calculate the prediction results of the model;
    Save the model weights with the highest accuracy;
end for
if training is over then
    Delete the classification layer of the model;
    Perform model conversion and save the .pt model;
end if
Implementation:
Load the .pt model on the Android platform;
Extract the features of the datasets images, and store them
in the database;
Extract the features of the query image;
Caculate the distance and get similar pictures.
return mAP.
```

4. Experiments

We conduct experiments to evaluate the performance of OMCBIR framework. These experiments are tested on five widely used datasets. Classification accuracy is an important indicator for evaluating the network structure. Therefore, we firstly evaluate the classification accuracy, computation, parameter amount and model size of ALNet and other classical lightweight neural networks. In addition, we compare mAP of ALNet with other state-of-the-art algorithms in OMCBIR. Finally, the ablation experiments of different components are performed, using MobileNetV2 as a baseline.

4.1. Datasets

MNIST [35] dataset is a very classic dataset in the field of image classification, consisting of 60,000 training samples and 10,000 test samples, each of which is a 28×28 pixel gray scale handwritten digital image. Since Pytorch Mobile does not support reading gray scale images, we use the method of overlaying channels for training.

SVHN [36] (Street View House Number) dataset comes from the Google Street View house number and contains more than 600,000 digit images. The native dataset is a number of unprocessed color images with multiple digits on each image. This dataset contains PNG images and digitStruct.mat files. The .mat files are used in the experiments, and each image is 32×32 in size. When performing image retrieval, .mat file is needed to convert into an RGB image by using the image's filename as GroundTruth.

SUN397 [37] dataset contains 108753 images of 397 categories, used in the Scene UNderstanding (SUN) benchmark. The number of images varies across category, but there are at least 100 images per category. We process the image categories into labels, then training images and test images are combined, de-duplicated and sorted. Finally, the training images and test images are divided into 8:2 according to random grouping. After our processing, there are 76,872 training images and 19,218 test images, and the total dataset contains 96,090 images of 397 categories.

Oxbuild5K [38] dataset contains 5063 images of 17 Oxford landmarks obtained from Flickr. We process the categories into labels, corresponding to 1-17 numbers respectively. We have preprocessed the image so that the image size for input model training and image retrieval is 160×160.

Paris6K [39] dataset contains 6412 images of 12 Paris landmarks obtained from Flickr. We perform the same data processing as Oxbuild5K on this dataset.

4.2. Evaluation Criteria

For the evaluation of the proposal, we use widely adopted evaluation metrics, namely the number of parameters and the required floating-point operations (FLOPs), to evaluate the memory occupation and calculation requirements of the model. We also use Model Size to measure the memory footprint of the model, which is the size of the saved .pt model. FLOPs can directly determine the inference time of the model, and if the complexity is too high, it will cause the model training and prediction to take too much time. Params and Model Size determine the memory footprint of the model, which is a crucial factor that affects the implementation of the algorithm model.

To compute the number of FLOPs, we only consider the flops of convolution operation, because other operations (such as batch normalization and pooling) are insignificant compared to convolution operations. As mentioned in [40], for convolutional kernels, we have:

$$FLOPs = 2HW(C_{in} \times K^2 + 1)C_{out} \tag{5}$$

where H, W and C_{in} are respectively the height, the width and the number of channels of the input feature map, K is the kernel width (assumed to be symmetric), and C_{out} is the number of output channels.

When calculating the parameters of the model (Params), we only consider the convolution operations. We use the most common calculation method, for convolutional kernels, defined by equation(4):

$$Params = K^2 \times C_{in} \times C_{out} \tag{6}$$

where K is the kernel width, C_{in} is the number of channels of the input feature map, and C_{out} is the number of output channels.

In order to evaluate the ability of the model on image retrieval tasks, we use mAP as an evaluation index of retrieval performance to measure retrieval results. mAP represents the average percentage of similar images in all retrieved images after evaluating all queries. The formula for the average accuracy of k searches is calculated as follows:

$$precision@k = \frac{\sum_{i=1}^{k} R_e(i)}{k} \tag{7}$$

where $R_e(i) \in [0, 1]$ indicates whether the current K-th image is an image with the same label and determines the correlation between the query image and the i-th image. 0 means that this image does not have the same label as the query image, and 1 means that this image has the same label. The average precision (mAP) formula is defined as follows:

$$mAP(Q) = \frac{1}{Q} \sum_{i=0}^{Q} \frac{1}{m} \sum_{k=1}^{m} precision@k \tag{8}$$

where Q is the number of the set of query images, and m is the number of similar images in the dataset.

4.3. Experimental Setting

For the experiments, Pytorch is used to train the classification model and Pytorch Mobile framework is used to complete the deployment of the deep learning model on the mobile terminal. The experiments use the Stochastic Gradient Descent (SGD) optimization algorithm, and the weight decay is 1×10^{-4}. For datasets with smaller images such as MINIST and SVHN, batch size during training is 128, and 64 during testing. For other datasets with larger size of images, batch size during training is 32, and 16 during testing. The initial learning rate is 0.1. According to the total number of training times, different epoch numbers are set to reduce the learning rate. In order to prove the effectiveness of ALNet, we have conducted comprehensive experiments on ALNet, ShuffleNet, ShuffleNetV2, MobileNets , MobileNetV2 and EfficientNet on the five public datasets. These experiments are divided into two stages. The first stage is model training, and we record model parameters after training, aiming at high classification accuracy, low computation and low memory consumption. The second stage is image retrieval, with mAP and mAP@5 as evaluation indicator.

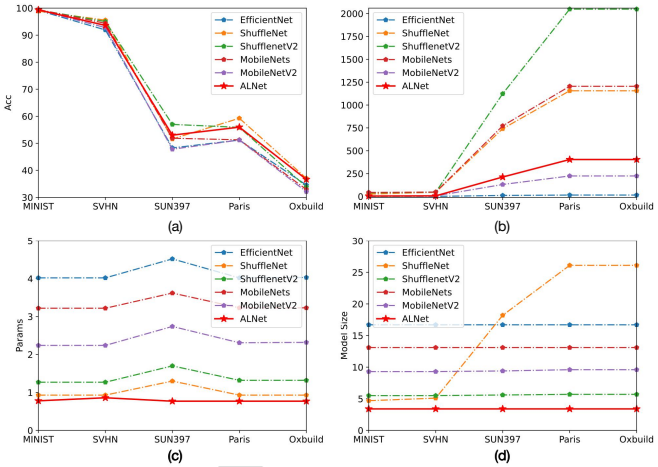

Fig. 3. Comparison of model performance on five datasets. The figure shows the comparison of the Acc, FLOPs, Params, and Model Size indicators of different algorithms on five datasets. From the results, it can be seen that ALNet shows excellent performance.

4.4. Evaluation on Model Metrics

In this subsection, we compare ALNet with classic lightweight neural network models (including ShuffleNet, MobileNets, ShuffleNetV2, MobileNetV2 and EfficientNet) in recent years. We conduct feature extraction training for the model oriented to classification tasks, so in this section we show the comparison of the Acc, FLOPs, Params, and Model Size of different algorithms on five datasets. As shown in Figure 4 (a), (c) and (d), the experimental comparison shows that the proposed ALNet is much smaller than other lightweight models in terms of Params and model size when there is little difference in classification accuracy, which can save a lot of memory footprint. It can be seen from Figure 3(b) that on all datasets, FLOPs of ALNet are much smaller than ShuffleNet, ShufflenetV2 and MobileNets. Although FLOPs on the latter three datasets is slightly higher than that of MobileNetV2 and Efficientnet, the performance is better on the other three indicators. From the analysis of the above results, ALNet still maintains a high classification accuracy while greatly reducing the complexity of the model, which verifies the effectiveness of our improvements on ALNet.

4.5. Evaluation on Retrieval Task

We use ALNet without the last layer of classifier as the feature extractor on the mobile terminal to test the retrieval performance of OMCBIR. For other popular lightweight CNN networks (including ShuffleNet, MobileNets, ShuffleNetV2, MobileNetV2 and EfficientNet), we use the same method to test retrieval performance on the mobile side. Moreover, we analyze the performance on each dataset and compared with other state-of-the-art methods. Finally, we conduct ablation experiments on the proposed ALNet in OMCBIR. The details of experiments are shown in the following.

Table 2. Comparision with The Five Methods on MINIST.

Model	FLOPs	Params	Model Size	mAP@1	mAP@5
MobileNets	44.04M	3.22M	13.1M	86.40	98.51
ShuffleNet	26.04M	0.93M	4.7M	27.03	98.51
ShuffleNetV2	39.29M	1.27M	5.5M	84.81	99.23
MobileNetV2	6.10M	2.24M	9.3M	83.91	99.14
EfficientNet	**1.09M**	4.02M	16.7M	21.67	93.64
ALNet	4.56M	**0.78M**	**3.4M**	**96.61**	**99.80**

Table 3. Comparision with The Five Methods on SVHN.

Model	FLOPs	Params	Model Size	mAP@1	mAP@5
MobileNets	48.24M	3.22M	13.1M	73.77	97.98
ShuffleNet	46.28M	0.93M	5.1M	71.94	97.77
ShuffleNetV2	47.25M	1.27M	5.5M	**84.66**	98.37
MobileNetV2	6.69M	2.24M	9.3M	78.60	97.91
EfficientNet	**1.27M**	4.02M	16.7M	75.74	97.96
ALNet	6.64M	**0.86M**	**3.4M**	82.66	**98.46**

Table 4. Comparision with The Five Methods on SUN397.

Model	FLOPs	Params	Model Size	mAP@1	mAP@5
MobileNets	772.27M	3.62M	13.1M	**37.12**	94.20
ShuffleNet	740.88M	1.30M	18.2M	32.07	94.98
ShuffleNetV2	1123.0M	1.70M	5.6M	35.34	95.13
MobileNetV2	131.15M	2.74M	9.4M	33.60	95.29
EfficientNet	**11.24M**	4.52M	16.7M	35.92	94.92
ALNet	212.79M	**0.77M**	**3.4M**	33.67	**95.76**

4.5.1. Results on MINIST

We analyze the performance on MINIST. As shown in Table 2, ALNet is far superior than other algorithms in terms of retrieval performance, mAP@1 is up to 96.61%, 10.21% higher than MobileNetV2. mAP@5 is 99.80%, which is also higher than other algorithms. At the same time, FLOPs, Params and Model Size have also reached the minimum. Among them, FLOPs is 4.56M, which is 25.2% lower than MobileNetV2 and nearly one tenth of the other three algorithms. Model Size is only 3.4M, which is convenient for deployment in smart phones or wearable devices that have high requirements for computation power and memory occupation.

4.5.2. Results on SVHN

Table 3 summarizes the comparison results of each method on SVHN dataset. The proposed method obtains a mAP@1 of 82.66%, second after ShuffleNetV2, but FLOPs, Params and Model Size are smaller than ShuffleNetV2, it is worth mentioning that the amount of calculation is only 14% of ShuffleNetV2. At the same time, compared with MobileNetV2, our method reduces FLOPs by 0.7%, Params by 61.6% and Model Size by 63.4% when mAP@1 is higher than the baseline. The comprehensive advantages are more prominent.

4.5.3. Results on SUN397

Experiments on the large-scale scene understanding dataset SUN397 are conducted and the experimental results are shown in Table 4. Our proposed method obtains the highest mAP@5 of 95.76%. Although slightly unsatisfactory on mAP@1, it is still higher than the baseline model. Compared with the first three methods, ALNet produces a better compression and acceleration rate than the ShuffleNet series and MobileNets, as shown in FLOPS, Param, and Model Size. Compared with MobileNetV2, although it is slightly higher in FLOPs, ALNet has advantages in other aspects. (In terms of mAP@1, they are 33.67% and 33.60%, in terms of parameters, they are 0.77M and 2.74M, and in terms of Model Size, they are 3.4M and 9.4M). Therefore, the proposed method demonstrates its high-performance retrieval capabilities and the ability to compress lightweight neural networks.

Table 5. Comparision with The Five Methods on Oxbuild5k.

Model	FLOPs	Params	Model Size	mAP@1	mAP@5
MobileNets	1205.0M	3.23M	13.1M	19.38	92.51
ShuffleNet	1156.0M	0.93M	26.1M	19.23	92.96
ShuffleNetV2	2049.0M	1.32M	5.7M	18.38	92.43
MobileNetV2	224.39M	2.32M	9.6M	18.03	93.09
EfficientNet	**15.65M**	4.03M	16.7M	19.56	89.65
ALNet	404.31M	**0.77M**	**3.4M**	**19.91**	**93.58**

Table 6. Comparision with The Five Methods on Paris6k.

Model	FLOPs	Params	Model Size	mAP@1	mAP@5
MobileNets	1205.0M	3.23M	13.1M	24.41	**94.25**
ShuffleNet	1156.0M	0.93M	26.1M	22.82	92.95
ShuffleNetV2	2049.0M	1.32M	5.7M	23.14	93.45
MobileNetV2	224.37M	2.31M	9.6M	22.57	92.60
EfficientNet	**15.64M**	4.02M	16.7M	23.60	92.10
ALNet	404.31M	**0.77M**	**3.4M**	**24.87**	91.58

4.5.4. Results on Oxbuild5k and Paris6k

Table 5 and Table 6 summarize the comparison results on the Oxbuild5k and Paris6k datasets. The proposed method obtains 19.91% mAP@1 and 93.58% mAP@5 on Oxbuild5k, which is higher than other methods. In terms of Param and Model Size, the best results are also obtained, which are 0.77M and 3.4M respectively, which is convenient for deployment in mobile devices. We observe that ALNet always has a slightly higher calculation amount than MobileNetV2 on datasets with larger size of images (SUN397, Oxbuild5k, Paris6k), which is the weakness of our method, but it is better than MobileNetV2 in the other three indicators. It can be seen from Table 6 that the proposed method shows more advantages in compression rate and retrieval. Similar results are observed on Paris6K, the highest mAP@1 of 24.87% is obtained on Paris6k, Params and Model Size remain the same as Oxbuild5k, reaching the smallest 0.77M and 3.4M. This verifies the effectiveness of OM-CBIR.

4.6. Ablation Study

In order to evaluate the effectiveness of ALNet, we conduct detailed ablation studies on the five datasets. As shown in Table 7, we use MobileNetV2 as the baseline and denote our model

Table 7. Ablation Experiments on Five Datasets.

Methods	Params	Model Size	mAP(%)				
			MINIST	SVHN	SUN397	Oxbuild5k	Paris6k
MobileNetV2	2.24 ~ 2.74M	9.3 ~ 9.6M	83.91	78.60	33.60	18.03	22.57
LNet	0.55 ± 0.01M	2.3 ± 0.1M	81.28	49.47	30.29	18.77	18.94
LNet+CS	0.55 ± 0.01M	2.3 ± 0.1M	92.76	74.70	31.29	18.79	20.75
LNet+CBAM	0.77 ± 0.01M	3.3 ± 0.1M	86.41	61.59	29.90	19.44	22.69
LNet+CS+CBAM (ALNet)	0.77 ~ 0.86M	3.4M	**96.61**	**82.66**	**33.67**	**19.91**	**24.87**

that uses grouped convolution and the new network structure as Lite-Network (LNet). On the basis of LNet, a lightweight network with channel shuffle (CS) and CBAM is the proposed ALNet. Table 7 reports the results of various parts of the ablation experiment of ALNet. Similar observations can be found in other datasets.

Channel Shuffle: From the comparison of Params and Model Size of MobileNetV2 and LNet, it can be seen that grouping convolution can effectively reduce the Params and Model Size of the model, and the minimum values are obtained on all datasets. However, mAP is almost lower than the baseline on each dataset, so the grouping operation weakens the performance of the model to a certain extent. Channel shuffle is proposed to solve the problem of feature communication between different groups. It can be seen from the third row of Table 7 that adding channel shuffle does not increase the parameters and size of the model, but improves the model retrieval performance to varying degrees.

CBAM: Channel Attention Module and Spatial Attention Module perform attention on the channel and space respectively. It not only saves parameters and computing power, but also ensures that it can be integrated into the existing network as a plug-and-play module in the architecture. As shown in Table 7, after adding channel shuffle, although the retrieval performance on the MINIST, SUN397 and Oxbuild5k datasets is better than the baseline, it is still slightly insufficient on SVHN and Paris6k. In order to improve the feature expression ability of the model, we introduce an attention mechanism to let the network learn to pay attention to key information. From the fourth row of Table 7, it can be seen that mAP is almost improved after adding CBAM with only a small increase in parameters and model size.

By adding channel shuffle after the 1×1 group convolution and adding a convolutional attention mechanism after the depth separable convolution, we obtain the best model. The placement order of the above modules is the best result obtained through a large number of experiments. Comparing the rows in Table 7, we can see that both channel shuffle and convolutional attention mechanisms can basically enhance the retrieval performance of the model. It can be concluded from the last row of Table 7 that under the combined effect of the two, our model has obtained the highest mAP that exceeds other algorithms on each dataset.

5. Conclusions

This paper proposes an offline mobile retrieval framework called OMCBIR. In this framework, we use lightweight CNN to accomplish feature extraction in order to deploy the deep learning model on mobile device. To further reduce model redundancy, we redesign the block structure. By reconstructing the model structure and introducing the convolutional attention module, an extremely lightweight small network architecture ALNet is designed, which greatly reduces the computational cost and memory consumption of the model while improving the retrieval accuracy. After that, the model is used as a feature extractor, which is converted and deployed to Android platform to achieve image retrieval. A comprehensive experiments are carried out in the experimental verification part. We evaluate retrieval performance and model compression indicators of ALNet and the latest SOTA methods on different datasets. In addition, we also conduct ablation experiments on each component of the model. Experimental results show that under the OMCBIR framework, ALNet can achieve the best performance compared to the refered methods.

6. Acknowledgement

This work is partially supported by Natural Science Foundation of China under Grant No. U20A20196, 61976192 and Zhejiang Provincial Natural Science Foundation of China under Grant No. LR21F020002.

References

[1] A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, Neural codes for image retrieval, in: European conference on computer vision, Springer, 2014, pp. 584–599.
[2] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE internet of things journal 3 (2016) 637–646.
[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.
[4] Z. Qin, Q. Zeng, Y. Zong, F. Xu, Image inpainting based on deep learning: A review, Displays 69 (2021) 102028.
[5] C. Bai, L. Huang, X. Pan, J. Zheng, S. Chen, Optimization of deep convolutional neural network for large scale image retrieval, Neurocomputing 303 (2018) 60–67.
[6] W. Zhou, K. Chen, A lightweight hand gesture recognition in complex backgrounds, Displays 74 (2022) 102226.
[7] Y. Kalantidis, C. Mellina, S. Osindero, Cross-dimensional weighting for aggregated deep convolutional features, in: European conference on computer vision, Springer, 2016, pp. 685–701.

[8] G. Tolias, R. Sicre, H. Jégou, Particular object retrieval with integral max-pooling of cnn activations, arXiv preprint arXiv:1511.05879 (2015).

[9] A.S. Razavian, J. Sullivan, S. Carlsson, A. Maki, Visual instance retrieval with deep convolutional networks, ITE Transactions on Media Technology and Applications 4 (2016) 251–258.

[10] A. Babenko, V. Lempitsky, Aggregating deep convolutional features for image retrieval, arXiv preprint arXiv:1510.07493 (2015).

[11] S. Shahriar, Gan computers generate arts? a survey on visual arts, music, and literary text generation using generative adversarial network, Displays 73 (2022) 102237.

[12] C. Bai, A. Zheng, Y. Huang, X. Pan, N. Chen, Boosting convolutional image captioning with semantic content and visual relationship, Displays 70 (2021) 102069.

[13] K.G. Dizaji, F. Zheng, N. Sadoughi, Y. Yang, C. Deng, H. Huang, Unsupervised deep generative adversarial hashing network, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 3664–3673.

[14] J. Song, T. He, L. Gao, X. Xu, A. Hanjalic, H.T. Shen, Unified binary generative adversarial network for image retrieval and compression., Int. J. Comput. Vis. 128 (2020) 2243–2264.

[15] C. Bai, H. Li, J. Zhang, L. Huang, L. Zhang, Unsupervised adversarial instance-level image retrieval, IEEE Transactions on Multimedia (2021).

[16] Y. Cheng, D. Wang, P. Zhou, T. Zhang, A survey of model compression and acceleration for deep neural networks, arXiv preprint arXiv:1710.09282 (2017).

[17] S. Han, H. Mao, W.J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, arXiv preprint arXiv:1510.00149 (2015).

[18] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient convnets, arXiv preprint arXiv:1608.08710 (2016).

[19] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2736–2744.

[20] Y. He, P. Liu, Z. Wang, Z. Hu, Y. Yang, Filter pruning via geometric median for deep convolutional neural networks acceleration, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4340–4349.

[21] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, N. Sebe, Binary neural networks: A survey, Pattern Recognition 105 (2020) 107281.

[22] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531 (2015).

[23] Z. Huang, N. Wang, Like what you like: Knowledge distill via neuron selectivity transfer, arXiv preprint arXiv:1707.01219 (2017).

[24] Y. Zhang, T. Xiang, T.M. Hospedales, H. Lu, Deep mutual learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4320–4328.

[25] T. Furlanello, Z. Lipton, M. Tschannen, L. Itti, A. Anandkumar, Born again neural networks, in: International Conference on Machine Learning, PMLR, 2018, pp. 1607–1616.

[26] T. Palaniswamy, Hyperparameter optimization based deep convolution neural network model for automated bone age assessment and classification, Displays 73 (2022) 102206.

[27] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258.

[28] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6848–6856.

[29] N. Ma, X. Zhang, H.T. Zheng, J. Sun, Shufflenet v2: Practical guidelines for efficient cnn architecture design, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 116–131.

[30] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).

[31] A. Howard, M. Sandler, G. Chu, L.C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al., Searching for mobilenetv3, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1314–1324.

[32] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 6105–6114.

[33] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1, arXiv preprint arXiv:1602.02830 (2016).

[34] S. Woo, J. Park, J.Y. Lee, I.S. Kweon, Cbam: Convolutional block attention module, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 3–19.

[35] Y. LeCun, The mnist database of handwritten digits, http://yann. lecun. com/exdb/mnist/ (1998).

[36] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning (2011).

[37] J. Xiao, J. Hays, K.A. Ehinger, A. Oliva, A. Torralba, Sun database: Large-scale scene recognition from abbey to zoo, in: 2010 IEEE computer society conference on computer vision and pattern recognition, IEEE, 2010, pp. 3485–3492.

[38] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Object retrieval with large vocabularies and fast spatial matching, in: 2007 IEEE conference on computer vision and pattern recognition, IEEE, 2007, pp. 1–8.

[39] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Lost in quantization: Improving particular object retrieval in large scale image databases, in: 2008 IEEE conference on computer vision and pattern recognition, IEEE, 2008, pp. 1–8.

[40] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, arXiv preprint arXiv:1611.06440 (2016).

Highlights:

- We propose a novel offline framework for content-based image retrieval on the mobile side, namely OMCBIR. As far as we know, this is the first time that a deep learning model is used for a completely offline mobile content-based image retrieval task.
- As the core of OMCBIR, we propose an extremely lightweight network architecture ALNet, whose role is to extract features from images. Besides, we also propose an ALBlock unit, which is the basic building block of ALNet. By introducing pointwise group convolution, channel shuffle and convolution attention module in bottleneck block, we not only improve retrieval accuracy, but also greatly reduce the computational cost and memory usage of the model.
- The experimental results on five public datasets demonstrate the effectiveness of our model. Compared to MobileNetV2, ALNet obtains a higher mAP on each dataset in OMCBIR with over 62% reduction in model parameters and over 63% reduction in model size.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: