2017

Robust Mobile Visual Recognition System: From Bag of Visual Words to Deep Learning

Dawei Li
Lehigh University

Robust Mobile Visual Recognition System: From Bag of Visual Words to Deep Learning

by

Dawei Li

A Dissertation
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Computer Science

Lehigh University
May 2017

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dawei Li
Robust Mobile Visual Recognition System: From Bag of Visual Words to Deep Learning

---

Date

Professor Mooi Choo Chuah, Dissertation Director, Chair
(Must Sign with Blue Ink)

Accepted Date

Committee Members

Professor Brian D. Davison

Professor Michael Spear

Professor Wei-Min Huang

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor, Professor Mooi Choo Chuah, for her generous support, guidance, patience, and encouragement throughout the PhD study. Professor Chuah taught me a lot from critical thinking, motivating new problem, scientific evaluation, presentation organization to seeing the big picture. I am very grateful that Professor Chuah provides me the freedom to explore many interesting research problems and she is always helpful both on stimulating research ideas and on providing favorable research facilities. Without her advice, I could not have finished this dissertation.

I am also very grateful to Professor Brian D. Davison, Professor Michael Spear and Professor Wei-Min Huang for providing excellent suggestions for my dissertation and serving as my committee members.

I greatly appreciate Dr. Theodoros Salonidis, Dr. Nirmit Desai and Dr. Bong Jun Ko from IBM Research, and Dr. Xiaolong Wang and Dr. Jon Currey from Samsung Research America for offering me internship opportunities where part of this dissertation was done. I would also like to thank my collaborators Dr. Deguang Kong and Jian Huang who I met and know during the internship. In addition, the internship experiences provide me the opportunity to learn how my research can impact the world from from the perspective of the industry. I also made many good friends, have learned a lot from them and benefited greatly.

I gratefully acknowledge the support from the research sponsors, including National Science Foundation through CSR grant 1217379 and CSR grant 1016296, as well as Nvidia for an equipment grant.

Many thanks to my colleagues and friends in Lehigh University who offered help to me and my family in various ways. I thank my fellow labmates in Lehigh WiNS Lab: Qinghan Xue, Wenbo Li, Xin Li, Ziyuan Qin, Li Tian, Xin Wang for discussion and help on a wide area of topics.

Finally and most importantly, I am most grateful of my family for their love, support, and sacrifices. My wife, Lei Zhang has been with me all the time during my

graduate study and her optimistic attitude toward life created a loving atmosphere in our home. My children, Jack and Karen have brought me the greatest joy and motivated me to finish this thesis with the greatest determination. I saved the final word of acknowledge for my parents, Tijin Li and Xiuxia Sun, for offering me unconditional love and belief.

To my family.

Contents

List of Tables

List of Figures

Abstract

With billions of images captured by mobile users everyday, automatically recognizing contents in such images has become a particularly important feature for various mobile apps, including augmented reality, product search, visual-based authentication etc. Traditionally, a client-server architecture is adopted such that the mobile client sends captured images/video frames to a cloud server, which runs a set of task-specific computer vision algorithms and sends back the recognition results. However, such scheme may cause problems related to user privacy, network stability/availability and device energy.

In this dissertation, we investigate the problem of building a robust mobile visual recognition system that achieves high accuracy, low latency, low energy cost and privacy protection. Generally, we study two broad types of recognition methods: the bag of visual words (BOVW) based retrieval methods, which search the nearest neighbor image to a query image, and the state-of-the-art deep learning based methods, which recognize a given image using a trained deep neural network. The challenges of deploying BOVW based retrieval methods include: size of indexed image database, query latency, feature extraction efficiency and re-ranking performance. To address such challenges, we first proposed EMOD which enables efficient on-device image retrieval on a downloaded context-dependent partial image database. The efficiency is achieved by analyzing the BOVW processing pipeline and optimizing each module with algorithmic improvement.

Recent deep learning based recognition approaches have been shown to greatly exceed the performance of traditional approaches. We identify several challenges of applying deep learning based recognition methods on mobile scenarios, namely

energy efficiency and privacy protection for real-time visual processing, and mobile visual domain biases. Thus, we proposed two techniques to address them, (i) efficiently splitting the workload across heterogeneous computing resources, i.e., mobile devices and the cloud using our MOCA framework, and (ii) using mobile visual domain adaptation as proposed in our collaborative edge-mediated platform DeepCham. Our extensive experiments on large-scale benchmark datasets and off-the-shelf mobile devices show our solutions provide better results than the state-of-the-art solutions.

Chapter 1

Overview

In this chapter, we provide an overview of this dissertation. We start by briefly reviewing some useful mobile visual recognition applications. Then we investigate the limitations of traditional mobile-cloud based vision systems. After that, we study two types of visual recognition methods and discuss the challenges of deploying them for mobile scenarios. Finally, we summarize our contributions and present the organization of this dissertation.

1.1 Mobile Visual Recognition

Nowadays, smartphones equipped with capable cameras enable people to generate new visual content (i.e., images and videos) anytime anywhere. According to a recent prediction, around 1.2 trillion photos will be be taken by mobile users worldwide in the year 2017 [1]. In addition, the increasing popularity of mobile social apps (e.g., Facebook, Instagram, Snapchat, Twitter, WeChat etc.) makes visual content sharing more convenient. The fact that mobile users are creating and receiving a

huge number of images/videos from time to time, makes mobile visual recognition a particularly interesting and important feature.

One very useful application is image based product search. For example, by recognizing a garment item in an image, an interested user can match it to the same item in an online shop [2][3]. For such applications, a database of product images is pre-built and indexed using extracted image features. By searching the database (e.g., finding k nearest neighbors using Euclidean distance), the system returns a short list of hyperlinks associated with product images most similar to the query. By clicking on a returned image, users can be redirected to its online shop. Such image retrieval based visual recognition technique is also being applied to other interesting mobile apps such as recognitions of plants in botanical gardens [4], artworks [5], landmarks [6] etc.

Mobile augmented reality (AR) apps [7] such as Pokmon GO [8] and Field Trip [9] have recently attracted a lot of attention. Mobile AR apps add overlaid components to object recognized in streamed video frames captured by mobile devices' cameras, and provide much richer and possibly interactive content on the physical world (e.g., Pokmon GO superimposes a Pocket Monster over a recognized surface). Meanwhile, the emerging smart wearable glasses such as Google Glasses have brought many more possibilities for future mobile AR apps. Novel AR services nowadays can provide augmented information on real-world objects including building landmarks in travel apps [9, 10], human face recognition in a conference meeting [11], supplement work instructions for maintenance and repair [12], museum exhibits appreciation instruction [13] etc.

There are many other interesting and useful mobile apps. Scene recognition

[14] can help mobile cameras to automatically select the best scene mode which will produce the best results possible. Mobile check deposit allows bank customers to deposit checks by recognizing information with mobile camera. Printed word translation [15] can translate a book in an unknown language in real-time. All those applications rely on certain computer vision algorithms over captures visual contents by cameras of mobile devices.

1.2 Cloud Computing and Their Limitations

Relatively large-scale mobile visual recognition problem (e.g., retrieval over a 1-million image database, or classification over 1000 object categories) is currectly solved using a mobile-cloud system architecture (e.g., Google Goggles [16], CamFind [17]), considering the complexity of advanced computer vision algorithms as well as the constraint computing resources on mobile devices. A mobile client sends captured visual content (i.e., images or video clips) to the cloud which runs the specified computer vision algorithms on the input and sends back the recognition result to the mobile client. This system design assumes that cloud servers have powerful processing units with huge amounts of memory and storage, and thus can generally run many computer vision algorithms in real-time and scale well with increasing number of user requests. Meanwhile, it simplifies the mobile client design, i.e., no matter what kind of mobile devices the user is using (whether Android, iOS, or Windows), all it needs to do is to send captured visual content over the Internet.

Although the mobile-cloud architecture have some benefits, there are some serious drawbacks:

- **User Privacy Leakage:** Visual content captured by mobile devices may contain sensitive information such as faces, medicines a user is taking, personal items. Such private personal information can be captured by the cloud service and be leaked or sold to many unrelated parties.

- **Networking Issues:** The Internet connection is not always available for mobile users and may be unstable. The system may cause large recognition latency and even be completely unusable in such scenarios. Meanwhile, sending large visual contents over the Internet using cellular links is financially expensive when Wi-Fi is unavailable.

- **Device Energy Consumption:** Battery energy is still a scarse resource for mobile device and most mobile phones need to be charged at least once per day. Though a mobile client doesn't need to run complex computer vision algorithms on the device, sending multimedia data over the Internet is energy inefficient especially when the data is transferred over the cellular network, e.g. 4G LTE [18].

Though there have been much on-going research on solving those problems individually, a complete solution may need breakthroughs in several different research aspects including security, networking, GPU hardware design and battery etc. Therefore, in this dissertation, we aim to design a robust mobile visual recognition system by considering the overall system design approach and may involve heterogeneous computing resources, e.g., mobile devices, Edge server as well as the cloud.

1.3 Detailed Research Challenges

It is not trivial to build robust mobile visual recognition systems especially when users would like to perform relatively large-scale recognition tasks. Here, we elaborate challenges faced in designing two major categories of visual recognition systems.

1.3.1 Bag of Visual Words Based Retrieval Systems

Image retrieval systems match a query image/video frame to an image database and returns the visually most similar image(s) and(or) associated information (e.g., the history of a landmark). One perfect solution to eliminate all the drawbacks of cloud-based systems discussed above would be to run the complete retrieval pipeline efficiently on the mobile devices. However, such a solution faces the challenges due to the limited computing, storage and energy resources of the off-the-shelf mobile devices. In this dissertation, our explore typical challenges that one will face when designing such a system using the popular bag-of-visual-word (BOVW) framework [19]:

- **Searching Space Reduction:** The key problem for real-time image retrieval is in designing an effective searching space reduction method, i.e., a method which can significantly reduce the number of database images that need to be compared to a query image. For mobile image retrieval, if the context of the application scenario (e.g., location) can be inferred, we may reduce the searching space to only images matching the current context. However, such a strategy must be applied conservatively so that the target ground truth image is not excluded. One can further reduce the searching space in a BOVW

framework by comparing a query image to database images that share common visual words. However, selecting a good visual vocabulary size is challenging as using a large vocabulary will consume too much run-time memory, while using a small vocabulary may not effectively reduce the searching space.

- **Indexing Memory Cost:** On-device image retrieval assumes an image database is pre-constructed and already indexed for future searches. The database construction complexity is a one-time cost that can be performed on powerful cloud servers. However, the size of the database index should be compact enough to fit within the memory size of mobile devices. Existing work solves this problem by directly reducing the number of local features (e.g., SIFT [20] or SURF [21]) extracted from database images [22, 23]. However, such an approach only works on images without complex background or occlusion objects, otherwise the recognition accuracy significantly drops. For example, an image of a landmark may be occluded by trees, vehicles or people. Image features detected on those objects are actually irrelevant to the landmark itself [6]. Thus, it is not easy to select a true representative and compact set of features for an image.

- **Re-ranking Efficiency:** Re-ranking algorithms have been proposed to significantly improve the final retrieval accuracy. Up to now, many existing work [6, 22–25] rely on features' geometric information to exclude outlier features using a method like RANSAC [26]. However, such schemes require that each database image maintains the geometric metadata, e.g., the feature coordinates, which significantly increases the memory overhead on a mobile device.

Recently proposed k-nearest neighbor (KNN) based re-ranking method compares the similarity of the query image's KNN with the KNNs of top returned database images, and promotes images whose KNN is more consistent to the query image [27, 28]. Though this method dose not require additional memory space, it is computationally expensive since processing each query actually involves searching multiple images, i.e., both the query image and the top returned images.

1.3.2 Deep Learning Based Systems

Deep learning based visual recognition systems feed an input image or video frames to a trained deep neural network (normally a convolutional neural network [29]) which achieves high accuracy on many traditionally hard visual recognition tasks such as large-scale object classification [29–31] and object detection (i.e., simultaneously locate the object and recognize what the object is in an image) [31–34]. However, one faces challenges if we want to apply deep learning based approaches in mobile scenarios:

- **Energy/Privacy Tradeoff:** State-of-the-art deep neural networks are composed of many neural network layers [31]. With powerful high-end GPUs, deep learning methods can still be executed in real-time on a cloud server. However, such design requires mobile users sending raw images over the Internet which may compromise users' privacy. Encrypting images can guarantee no privacy information is leaked during the transmission, but in most scenarios, encrypted images need to be decrypted before they are being fed into a deep learning algorithm but often cloud servers are untrusted. On the other hand,

researchers have recently proposed methods to accelerate the execution of deep learning algorithms [35–37] so that it can be deployed entirely on a mobile device. However, the energy cost for processing each image is still considerably high even with those accelerated deep learning models (up to 50x more than a cloud-based system).

- **Visual Domain Bias:** A visual domain is the context or the environment where the deep learning model is applied to. Each domain (e.g., a library or a museum) has a subset of objects and has unique attributes such as lighting condition, background etc. The visual domain bias phenomenon means that trained deep learning models have bias on different visual domains, and the actual accuracy obtained can be dramatically different. Thus, visual domain adaptation is necessary for high-accuracy recognition during deployment[38]. Existing work solves this adaptation issue by fine-tuning a trained deep learning model [39, 40] on the subset of objects in a target visual domain, but such fine-tuning requires dedicated GPU servers and well-labeled high-quality training images. For mobile visual recognition in the wild, the possible number of different visual domains can be endless, and therefore it is really challenge to adapt a deep learning model in every visual domain.

1.4 Contributions

In this dissertation, we discuss how to design robust mobile visual recognition systems and our discussion focuses on two major types of recognition methods, namely

bag of visual words (BOVW) based image retrieval approaches and deep learning based visual recognition approaches. We design and implement novel systems which address most of the challenges discussed above. We also demonstrate that our proposed systems achieve better performance than the state-of-the-art systems via extensive system evaluations:

- For the bag of visual words based image retrieval systems, we propose and implement EMOD, an efficient image retrieval system which performs on-device image retrieval on context limited image database. EMOD adopts our new algorithm design and leverages the parallel computing capabilities made possible by multi-core mobile processors. Our EMOD prototype system on Android platform achieves real-time high-accuracy retrieval with significantly reduced memory and energy consumption.

  - We address the problem of indexing memory cost by proposing an object word ranking algorithm to identify the most useful visual words of a database image, so that each image can be represented using a concise signature with 85% memory saving.
  - We propose a top-inverted-index ranking method that applies to the bag-of-visual-word framework to reduce 95% of searching space for real-time on-device image retrieval.
  - We propose a novel unsupervised 2-stage k nearest neighbor (KNN) based re-ranking method. Our algorithm outperforms state-of-the-art re-ranking algorithms on several benchmark datasets with minimum memory cost and reduced time latency.

- For deep learning based systems, we propose MOCA an energy-preserving and privacy-preserving mobile visual recognition system. It decouples the hierarchical layers of DNN models and splits them across mobile and cloud. It uses privacy-preserving feature maps extracted from DNNs on mobile as the input for further deep processing on cloud to protect personal user data.

  - We propose five novel techniques for improving the energy efficiency of the feature map extraction process on mobiles which include layer decomposition, parameter-free layer merging and branch reduction. Experimental results with a variety of deep learning based mobile applications demonstrate that the privacy of these data are well preserved, while the energy overhead on mobile device is 1.2x - 6.7x (16.67% - 85.07%) [1] lower than the state-of-the-art approach.

  - We propose a deep learning based framework that allows mobile application developers to quantify the privacy level for user data based on their own privacy criteria. We present the differentiated privacy mechanism for DNN-based mobile applications, and enable adaptive DNN model reconstruction to utilize mobile resources and achieve better privacy guarantee.

- We propose an edge-mediated framework, DeepCham, to adapt deep learning models for object recognition under different mobile visual domains. DeepCham generates high-quality domain-aware adaptation instances from crowd-sourced

[1] Here is how the values are calculated and converted: suppose that system A has energy cost e_a and system B has energy cost e_b, if e_b/e_a=6.7, we say that system A's energy overhead is 6.7x lower than system B and or system A's energy overhead is 85.07% lower than system B which is calculated as $(e_b\text{-}e_a)/e_b$.

in-situ mobile photos, and the adaptation can be agilely executed in real-time without dedicated GPU server.

  - We create a new image dataset, Mobile Context Image Dataset (MCID), which enables controlled experimental evaluation of deep model adaptation of different mobile visual domains. Using MCID, We show that DeepCham achieves as much as 150% accuracy improvement over the baseline deep learning model without an adaptation.

We now discuss the contributions in detail:

We first study the image retrieval problem using the popular bag-of-visual-word framework. Our proposed EMOD system identifies the potential for reducing runtime memory cost by analyzing the database image indexing pipeline. To construct a concise representation for a searchable object, it is critical to identify the most salient features or visual words of such object. We address this problem by proposing an object work ranking algorithm which ranks the importance of a visual word for a target object using multiple wild-captured images each from a different point of view. We show that our method greatly reduces the size of database image signatures, but also increases the retrieval accuracy due to the filtering of less important or noisy visual words. Then we analyze the potential for search space reduction by analyzing the query image searching pipeline, and propose a novel method to ignore database images with few common visual words to the query. In addition, we propose an efficient 2-stage KNN based re-ranking algorithm: we locate potential low-ranked ground-truth images in the first step and compare the KNN similarity in

step two using a new KNN similarity measure. For computation efficiency, we pre-calculate the KNN of all database images which are loaded in memory. Compared to geometric based re-ranking method, we achieve up to 90% memory saving, while we show that the retrieval accuracy after re-ranking outperforms the state-of-the-art on several benchmark datasets (including a large 1 million image dataset). Finally, we analyze the parallelism in the query pipeline by leveraging multiple-core processors of modern mobile devices for further speed-up. Experimental results show that real-time and high-accuracy retrieval over a large image database is achieved while the memory and energy overhead has negligible impact on off-the-shelf mobile devices.

For deep learning based visual recognition, our first contribution is, MOCA, which aims for energy-efficient and privacy-preserving deep processing for mobile visual recognition. Recently an increasing number of mobile vision applications are demanding deep learning techniques to empower intelligent services, such as face recognition and object detection. However, it is still unattractive to run these computationally intensive services on resource-constrained commodity mobile devices. Offloading these services to cloud reduces the burdens on mobiles, but may expose users sensitive data to untrusted cloud servers. MOCA decouples the hierarchical neural network layers and splits them across mobile and cloud for energy-efficiency. MOCA leverages the privacy-preserving feature maps extracted from deep models to protect personal data privacy. MOCA also enables differentiated privacy settings via a DNN-based privacy quantification framework (under the threat model we defined in chapter 3) to meet the varying privacy requirements of different mobile applications. Our experiments with various mobile vision applications against thousands of photos and videos from a variety of validation datasets demonstrate that MOCA

has up to 6.7x (85.07%) lower energy overhead on mobile devices compared to the state-of-the-art approaches, while providing stronger data privacy with negligible impact on application accuracy.

Finally, we present DeepCham, a crowd-sourced deep learning model adaptation framework. DeepCham is the first piece of work that solves the mobile visual domain bias problem using a trained large-scale deep learning model. The proposed adaptation method does not require high-end GPU servers or expert-created high-quality training images. Instead, DeepCham leverages abundant metadata-rich (e.g., GPS location, time) photos that have already been taken by mobile users, and trains a shallow adaptation model (e.g., SVM) in real-time on a mobile device. To achieve this, DeepCham includes an attribute-based visual domain specification module that identify photos matching a target visual domain, an efficient object bounding box proposal method to locate objects shown in a photo, and an interactive object labeling pipeline which suggests labels for an input image using a generic deep learning model. Our experiment on a new collected Mobile Context Image Dataset (MCID) obtains as much as 150% accuracy improvement. Our prototype system shows that all modules can run on a mobile device so that the user privacy is well protected. In addition, we demonstrate that our system is more energy efficient than cloud computing solutions.

1.5 Organization

This dissertation is organized as follows.

In Chapter 2, we provide an overview of the bag of visual words (BOVW) image

retrieval framework, and present our EMOD system [41] for efficient image retrieval. EMOD includes three novel algorithms: the object word ranking algorithm to reduce the indexing memory cost of database images, the top-inverted-index ranking algorithm to reduce the query searching space, and the light-weight 2-stage KNN re-ranking method to increase accuracy. We further parallelize the image query pipeline with multi-threaded processing on multi-core mobile processors and achieve better efficiency.

In Chapter 3, we study the problem of optimizing the energy cost and the privacy protection of deep learning based visual recognition, and present our MOCA system. MOCA uploads only privacy-preserving visual features extracted from users data to the cloud, while utilizing the cloud for high-performance deep processing. To accelerate the feature extraction procedure and reduce the feature size, several optimizations including layer decomposition and parameter-free layer merging are proposed and implemented in MOCA. It also enables differentiated privacy for different mobile applications.

In Chapter 4, we study the problem of mobile visual domain adaptation in deep learning based mobile visual recognition system, and present our DeepCham framework [42], the first adaptive mobile object recognition framework that allows deep learning model to be used successfully in mobile environments. Specifically, DeepCham is mediated by an edge master server which coordinates with participating mobile users to collaboratively train a domain-aware adaptation model which can yield much better object recognition accuracy when used together with a domain-constrained deep model.

Chapter 5 concludes the dissertation, summarizes what have been learned, and

discusses future work directions.

Chapter 2

EMOD: An Efficient On-device Mobile Visual Search System

Recently, researchers have proposed solutions to build on-device mobile visual search (ODMVS) systems. Different from traditional client-server mobile visual search systems, an ODMVS supports image searching directly within a mobile device. An ODMVS needs to be designed with constrained hardware in mind e.g. limited memory, less powerful CPU. In this chapter, we present, EMOD, an efficient on-device mobile visual search system based on the Bag-of-Visual-Word (BOVW) framework. Based on users' mobile context (e.g., location), EMOD downloads an indexed partial image database to the mobile devices for on-device image retrieval. An Object Word Ranking (OWR) algorithm is proposed to efficiently identify the most useful visual words of an image so as to construct a compact image signature for fast retrieval and greatly improved retrieval performance. Due to having a small

visual dictionary, we propose the Top Inverted Index Ranking scheme to reduce the number of candidate images for similarity calculation. In addition, we propose an efficient 2-stage k-NN based re-ranking method as an essential final step to improve the retrieval accuracy. Via extensive experimental evaluations, we demonstrate that our prototype EMOD system yields better retrieval accuracy and query response time than the state-of-the-art for a database with over 10K images.

2.1 Introduction

Visual search application allows a user to obtain information about objects in his proximity by taking an image of the object and using this to search a database of known objects. Such applications have become very popular among mobile users as mobile devices become more powerful with advancements in processing power, screen size and availability of higher wide area wireless bandwidth. Today, mobile visual search is available through services such as Google Goggles [16] or CamFind [17]. Such mobile visual search systems allow users to recognize visual objects in their vicinity. Interesting applications include recognitions of CD covers [43] or products [17] for online shopping, recognitions of plants in botanical gardens [4], artworks [5], or landmarks [44]. In addition, such image search applications are also very useful for government organizations e.g. criminal suspect identification [45], disease detection and diagnosis [46].

Traditionally, the mobile visual search systems [43][44] [47][5] adopt a client-server (CS) architecture. A user uses his mobile device to send a compressed image or an image signature extracted from the captured image to a remote server

which performs visual search related operations on an image database. However, such cloud-based systems typically face performance related issues either due to insufficient mobile network coverage or the inability of the recognition engine to process large quantity of simultaneous image matching quickly against a large image database. Such performance issues can potentially hamper the adoption rate of emerging mobile visual search applications e.g. a scan-and-purchase clothing application.

With the advances in mobile devices, on-device mobile visual search systems (ODMVS) have been recently proposed to solve the above problems [48][22][49]. Compared to the client-server architecture, ODMVS has several benefits such as network independence, better privacy control, and low query latency. Meanwhile, ODMVS can be integrated to an existing client-server based system. For example, a landmark recognition application allows a user to download a subset of a large image database, e.g. based on his location information, in the presence of a good quality network connection (e.g. WiFi). Then, a user can perform on-device visual search whenever he wishes without having to contact the remote server.

Due to the hardware limitations of mobile devices, more efficient image processing and matching techniques need to be designed for ODMVS. In this chapter, we present the design and evaluation of an efficient on-device mobile visual search system (EMOD) based on the popular *Bag-of-Visual-Word* (BOVW) framework [50]. Our EMOD design includes (1) pipelining the image database construction to reduce its memory impact, and (2) pipelining the image query process to reduce the query processing latency. Efficient techniques are proposed to identify (a) compact image signatures during image database construction, and (b) images which match

a query image efficiently. Such techniques ensure that EMOD yields satisfactory retrieval performance with low latency.

In summary, EMOD makes the following contributions:

- designs an efficient ODMVS which uses optimized Bag-of-Visual-Word framework with improved retrieval performance, and lower resource cost.
- uses our proposed *Object Word Ranking* algorithm to identify the most useful visual words of a database image so that each image can be represented using a concise signature with 85% memory storage saving,
- uses a modified inverted index scheme, named *Top Inverted Index Ranking*, to reduce the number of candidate images, and hence improves the query response time,
- uses our proposed efficient 2-stage re-ranking method to re-rank retrieved images for improved retrieval accuracy,
- achieves low query response time by exploiting the parallel computing capabilities made possible by mobile devices with multi-cores.

The rest of the chapter is organized as follows: we introduce the background of the Bag-of-visual-word framework in section 2.2 and review the related work; the system overview and our design principles are presented in section 2.3. This is followed by a detailed system algorithmic design in section 2.4 and the 2-stage k-NN re-ranking algorithm in section 2.5. We present the extensive system evaluations in section 2.6.

2.2 BOVW Background and Related Work

The Bag-of-visual-word (BOVW) framework was proposed [50] to effectively reduce the storage and computation overhead of a visual search system. The visual words are centroids of clusters formed using feature descriptors, e.g. SIFT [20] and SURF [21] extracted from training images. Typically, a very large visual dictionary, e.g. 1M visual words, is required to achieve the best retrieval performance. Once a visual dictionary is constructed, the feature descriptors from a single image are then quantized into a word histogram, i.e. each descriptor is classified to the nearest word in the dictionary. When a very large dictionary is used, the resulting word histogram of an image is a very sparse vector, and thus one can construct a compact signature for each image, i.e. only words with frequency greater than 0, and their index are kept. With such a representation, the signature of an image with 1000 SURF descriptors only consumes fewer than 5 KB. Next, a selected weighting scheme, typically based on the *tf-idf* of visual words, is applied to an image signature, and the similarity between two images are then computed as the distance (e.g. Euclidean or Cosine distance) between the two weighted image signatures. To speed-up the matching process, an inverted index is built in which a visual word is linked to a list of images containing this visual word. When a query image is received, the system identifies database images with common visual word(s) as the query image as candidate images. With a large visual dictionary, normally only a small number K of candidate images (i.e. the candidates) need to be compared with the query image, which significantly reduces the computational overhead. Finally, the top N_r of the ranked candidates are re-ranked, for example, using the geometric information associated with the raw image features, and the re-ranked image list is returned.

Any real-time visual search system built using the BOVW framework needs to efficiently perform feature quantization, i.e., to classify an extracted feature descriptor to the nearest visual word in the dictionary. For a large visual dictionary with 1 Million visual words, each image feature has to be compared against each visual word in the dictionary to find a nearest neighbor. Two main approaches are used to speed up this feature quantization process: the hierarchical k-means (HKM) approach using a visual vocabulary tree [51] and the approximate nearest neighbor (ANN) approach using a plain dictionary [52]. We will discuss both schemes in detail in section **2.4.1**.

ODMVS requires compact signatures to represent the data-base images. Thomee et al. [23] proposed to use only the most frequent or top weighted words, but using a large dictionary means that the majority of visual words has only a few counts and the noisy words cannot be detected ahead of time. Turcot et al. [53] and some follow-up work [54][44] suggest only keeping useful features in a database image. Such features are determined by matching against other database images containing the same object. Meanwhile, a word augmentation method is introduced to incorporate words from other images containing the same object into the signature of an image considering that each image shows a different view of the object. However, two factors can affect the performance of this approach under the BOVW framework. First, as explained in [55][52], matched features can be quantized into different words and hence such "falsely" matched features cannot be detected. Second, the number of matched features varies greatly for each database image which results in varying image signature length. Such variation has a negative impact on the retrieval performance. In this chapter, we propose the "Object Word Ranking"

algorithm to solve these two problems (see section **2.4.2**).

ODMVS can only afford a relatively small visual dictionary so that the entire dictionary can be loaded into the system memory. Hartl et al. [22] proposed to reduce the 64-value SURF descriptor into 32 values so that the visual dictionary can be reduced by half. However, the retrieval accuracy using this approach significantly reduced. Chen et al. [49] adopted a variant of the BOVW scheme, which generates an image signature using a residual visual vector instead of using the raw word histogram. This scheme performs well using a 128-word visual dictionary, and each database image can be represented with a signature of 533 bytes. However, with such a small dictionary, it cannot take advantage of the inverted index to reduce the number of candidate images for similarity calculation, and thus yields to relatively large query latency. In contrast, EMOD uses a larger 30K visual dictionary, and we propose a modified inverted index scheme to speed up the query processing (see section **2.4.3**).

Re-ranking schemes are proposed to improve the final retrieval performance. Up to now, many existing work [49][22] [44][52][53] uses RANSAC [56] to verify the matched features geometrically so that images with more inlier matches are promoted to a higher rank. However, this scheme requires that each database image maintains the geometric metadata, e.g. the feature coordinates, which significantly increases the memory overhead. In this chapter, we propose an efficient 2-stage KNN based re-ranking *Ranking Consistency* [57] method, which proves to be as powerful as the geometric verification method but consumes considerably less memory space (see section **2.4.4**).

Various compression and encoding methods are proposed to the BOVW framework [48][58][59][60] to further reduce the memory impact. Those methods are not discussed in this chapter, but can be easily integrated into our system.

2.3 System Overview

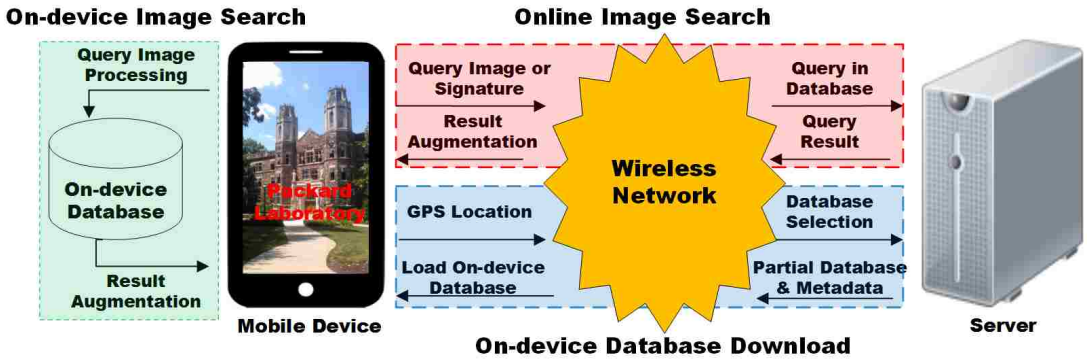


Figure 2.1: EMOD System Overview

Figure 2.1 illustrates the overview of our EMOD system, which allows users to operate with (online mode) and without network access (offline mode).

In the online mode, a user sends image queries to a remote server just like those traditional client-server (CS) mobile visual search systems. The received query result can be displayed on the screen of that user's mobile device or augmented over the query image. Besides the image retrieval function, our system allows users to download a subset of image database according to the context of the mobile application (e.g., based on a pre-specified geographical city region), when a suitable network connection is present (e.g., a good Wi-Fi connection) so that users can operate in an offline mode.

The offline mode can be manually activated by a user or automatically activated when a network connection is slow/lost. In the offline mode, the mobile device first loads the downloaded image database as well as its associated metadata (e.g. the visual dictionary, the inverted index etc.) to the memory. When a query image is captured, it is processed locally on the mobile device, and any query result can then be displayed directly on that device.

Here, we focus our work on the design of the offline mode feature, and our goal is to achieve real-time image query using a 10K image dataset on the mobile device with decent retrieval performance and minimal memory cost.

2.3.1 Design Principle

Figure 2.2 illustrates the our algorithmic optimization of EMOD over the BOVW framework. The design of EMOD adheres to the following two principles:

(1) *As small as possible*: given an image dataset, we want the system to consume as small memory cost as possible. With the BOVW framework, we aim to reduce the size of the constructed image database(all orange blocks shown in Figure 2.2). Meanwhile, we must maintain the retrieval performance at a good level.

(2) *As fast as possible*: given a query image, the system should return the query result as fast as possible. With the BOVW framework, we aim to reduce the computing latency of each component in the image retrieval pipeline (the yellow blocks shown in the right in Figure 2.2). We must also ensure that the speed-up does not significantly degrade the retrieval performance.

2.3.2 Database Construction Pipeline

The image database is typically constructed in an offline manner, and it includes (i) the visual dictionary, (ii) the image signatures,(iii) additional supporting data structures (e.g. the inverted index, the IDF table), and other useful metadata.

Visual Dictionary: Considering the memory limitation of mobile devices, we must use a smaller dictionary of size W than those (e.g. 1M) used in the server-side image search engines. In this chapter, we show that we are able to build an efficient ODMVS with a $W = 30$K dictionary. The 30K dictionary consumes only 7.5 MB memory which is acceptable for most mobile devices.

Image Signature: We proposed an *Object Word Ranking* (OWR) algorithm to select the N most useful visual words per image. Thus, each database image signature can be represented using two vectors of N elements: one for storing the visual word indices and the other for storing the frequency of such visual words in that particular image. With *N=100*, an image signature only consumes 300 bytes. The evaluation result shows that such compact signature is more representative of the image than all quantized visual words from the raw signature, and hence improve the retrieval performance of our EMOD system. (Details in section **2.4.2**)

Inverted Index Table: This table is used to generate the Top K candidate image list during query processing. It contains entries of visual words and their associated lists of image identifiers which contain that visual word. The table size is a function of the number of visual words in each image signature *N*. When *N=100*, each database image adds only 200 bytes (assuming each image identifier is a short integer) to the memory consumed by the inverted index table.

IDF: The IDF is an array of weighting values for each visual word in the dictionary. For the 30K visual dictionary, this table only consumes 117 KB memory.

Metadata: The metadata of each database image is used to re-rank the returned image list so that images containing the same object can be promoted to a higher rank. The traditional geometric verification-based re-ranking approach requires each image to store the geometric information (e.g. the coordinates) of each detected feature. However, even with the most advanced compression scheme [61], each feature still requires 6 bytes additional memory, and an image with 1000 detected features requires almost 6KB just to store such metadata. For efficient re-ranking, our system adopts a novel KNN based re-ranking scheme. For a database of V images, each image only needs to maintain a list of h=*0.005*V* top ranked image identifiers by using that database image as a query. (Details provided in section **2.4.4**)

2.3.3 Image Query Pipeline

Each image query requires real-time response, and hence we aim to reduce the computing complexity of each processing component in the image query pipeline. In this work, we do not attempt to modify the feature detection [62] and descriptor calculation step [21][20] but adopts the de facto standard scheme as used in [44][52][49][57][50][23].

Feature Quantization: To achieve real-time feature quantization, we adopt the fast approximate nearest neighbor [63][52] approach. The computing complexity to quantize each feature is reduced from O(W) (for brute-force nearest neighbor search) to O($\log W$), where W is the size of the visual dictionary. (Details in section

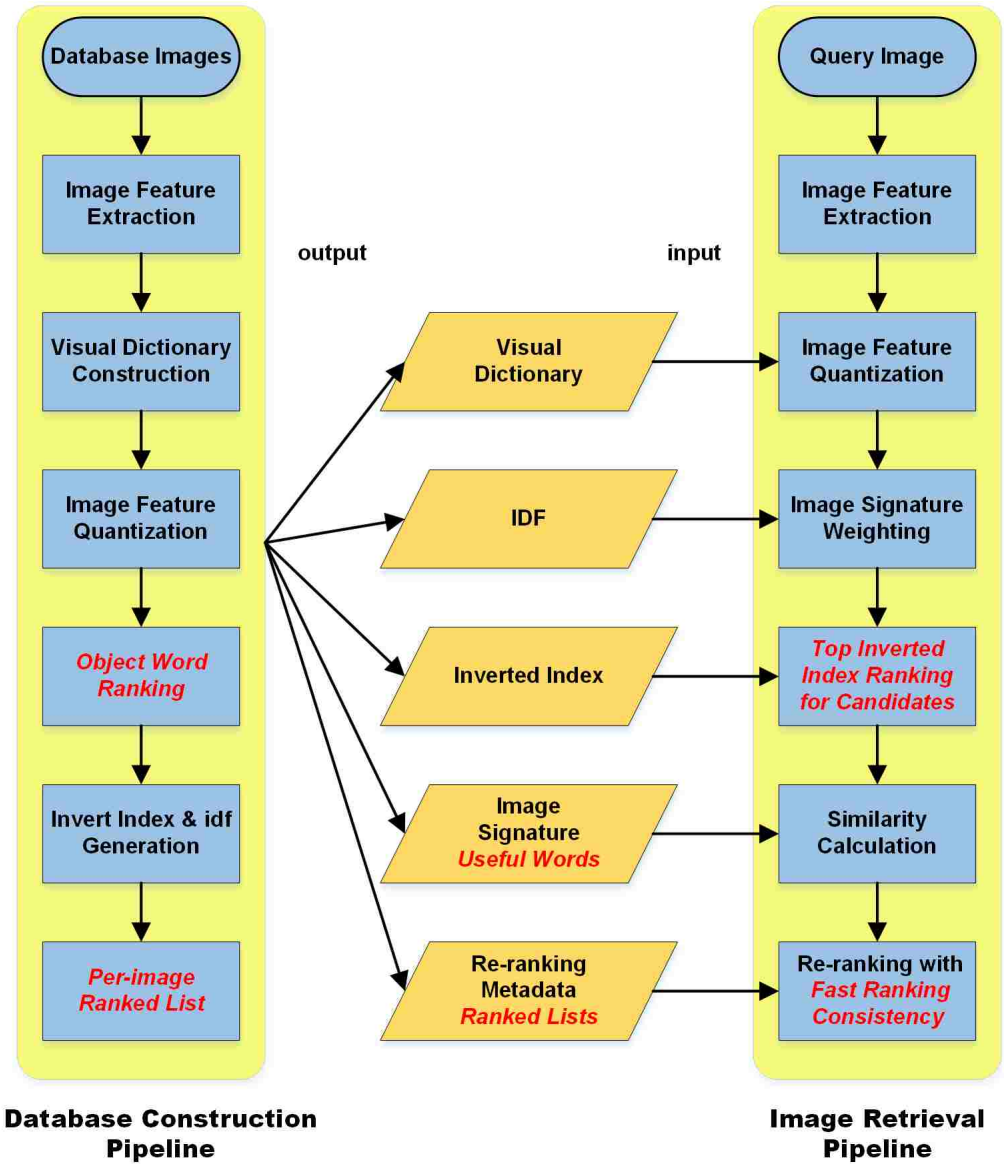


Figure 2.2: The Efficient On-device Mobile Visual Search System. The unique processing components are highlighted as red italics.

2.4.1)

Candidates Generation: The baseline BOVW system generates candidates images using the inverted index in a naive way: all database images containing visual

word(s) common to the query image are treated as *Candidates*. However, with a small visual dictionary and thousands of database images, too many *Candidates* are generated for image matching. In this work, we use our *Top Inverted Index Ranking* method to generate the top K database images with the most common words to the query image as *Candidates*, and thus achieve a fast image matching. (Details provided in section **2.4.3**)

Similarity Calculation: The similarity between two images are calculated as the cosine distance of two normalized *tf-idf* weighted signatures. Here, there is a tradeoff between the system memory cost, and the query processing speed as to whether the normalized *tf-idf* weighted signature of database images should be pre-computed or be calculated in real-time. The pre-computed signature would consume 300 more bytes per image for storing such weights. However, as shown in our performance evaluation, the extra memory cost only brings a slight decrease in query latency. Therefore, in this chapter, we compute the weights of the database image signature in real time for significant memory saving.

Result Re-ranking: In this work, we use the *Ranking Consistency* scheme for efficient result re-ranking. Instead of using the iterative re-ranking method proposed by the authors in [57], we use a simplified re-ranking method that substantially reduces the re-ranking latency but only has a slight degradation on the re-ranking performance. (Details in section **2.4.4**) Furthermore, we propose an efficient 2-stage re-ranking scheme for greatly improved re-ranking accuracy. In stage one, we generate an expanded list of candidate database images to re-rank so that some lower ranked ground truth images will not be ignored for the next stage. In stage two, we re-rank the list of candidate images using a confidence score which is calculated

based on both ranking consistency and reciprocal k-NN property. (Details in section **2.5**)

In addition, since modern mobile devices tend to have more CPU cores, the query processing components such as feature quantization and similarity calculation can be parallelized to speed up such computation steps. We discuss the performance improvement of parallelism in the system evaluation section.

2.4 Detailed Algorithmic Design

In this section, we provide more detailed description of our algorithm optimization that solves the following problems: visual dictionary construction, useful visual word selection, efficient image matching and re-ranking. With our proposed techniques, EMOD only consumes 13 MB for a database of 10K images. Real-time image query response time is also achieved with good retrieval performance.

2.4.1 Visual Dictionary Construction

In the BOVW framework, a visual dictionary is used to quantize image features into individual visual words which are later used for similarity calculation. When used in a real-time visual search system, the quantization is desired to be processed as fast as possible. Up to now, there are two main trends for fast feature quantization: the visual vocabulary tree (VVT) approach, and the fast approximate nearest neighbor (ANN) approach using a plain dictionary.

Visual Vocabulary Tree: Nister et al. proposed the popular *Visual Vocabulary Tree* (VVT) scheme [51], which is a hierarchical data structure constructed by

clustering the training feature descriptors with hierarchical k-means algorithm. A VVT can be defined as a B by D tree, where B is the branching factor, and D is the depth of the tree. The VVT is built iteratively: (1) all training descriptors are initially clustered into B clusters; (2) descriptors in each cluster are further clustered into B sub-clusters; (3) the process continues until depth D is reached. In [51], Nister et al. state that both leaf nodes and internal nodes can be regarded as visual words. However, Philbin et al. [52] show that using only leaf nodes results in better retrieval performance than including internal nodes as visual words. Therefore a 1M visual dictionary can be represented using a *B10D6* VVT. For feature quantization, one searches from the root to the leaf node level of the tree to find the best leaf node as the quantized visual word of every feature. This search process finds a nearest neighbor at each tree level and searches only the sub-tree of that nearest neighbor. Therefore, the quantization complexity for a feature is $\mathrm{O}(B * D)$ on a single CPU core, and it requires only 50% additional memory than the plain dictionary containing the same number of visual words. VVT is adopted or compared in some prior ODMVS related work [22][49].

Fast Approximate Nearest Neighbor on a Plain Dictionary: A plain visual dictionary of size W is created by clustering the training descriptors to W clusters, and each cluster centroid is regarded as a visual word. For feature quantization, each feature descriptor finds its nearest neighbor in the dictionary, which can be done quickly using fast approximate nearest neighbor methods. The most commonly used fast ANN method is the randomized *kd*-trees [64]. In the proposed scheme, a query data (i.e. the feature descriptor) is searched simultaneously in n randmoized *kd*-trees in a *Best Bin First* [65] manner. The search in each *kd*-tree

stops when i leaf nodes are reached, and the best candidate is returned. With properly selected n and i values, the query performance can be optimized [63], i.e. achieving a balance between query time and error rate. In a *kd*-tree, the query complexity is determined by the number of nodes explored for similarity computations (typically Euclidean distance calculations), n_d. Therefore with n trees, the quantization complexity for a feature is bounded by $O(n_d * n)$ on a single CPU core. However, with the "Early Termination" technique proposed by Keogh [66] [1], the actual quantization complexity is much smaller. In our experiment, we found that the quantization complexity of the randomized *kd*-trees scheme is approximately bounded by $O(i * n)$, i.e. the number of leaf nodes reached in each tree multiplied by the number of trees. On the other hand, additional memory is required for each *kd*-tree which is constructed using all the words in the visual dictionary.

Scheme Selection Criterion:In [52], Philbin et al. pointed out that VVT data structure generally produces more quantization errors than the plain dictionary using the randomized *kd*-trees scheme. However, they did not compare CPU and storage cost of these schemes. To determine which scheme we should use, we conduct an experiment using both schemes where we constrain the amount of memory cost for storing the visual dictionary, and the amount of CPU cycles for performing feature quantization. Then, we select the scheme which yields a better retrieval performance. We present our controlled comparison results in the system evaluation section.

[1] The "Early Termination" technique is also applied to the VVT data structure for finding the nearest neighbor in each level. However, only very limited speed-up is observed when the number of candidates in each tree level is small.

(a) All Detected Features (b) Selected Useful Features

Figure 2.3: Features detected on an image of the Memorial Church contains many noises from trees, cloud etc.

2.4.2 Useful Word Selection

A small vocabulary dictionary may lead to degraded retrieval performance. To compensate for this accuracy loss and to reduce the memory cost of image signatures, we propose the *Object Word Ranking* (OWR) method to retain only useful words of an image. Before describing OWR, we first describe the "Useful Feature" scheme proposed in an earlier work and its limitations. We also suggest a simple improvement to this scheme. In this section, we regard images containing the same object as *friend* images to ease our explanation.

"Useful Feature" and The Problems

Photos taken by a mobile device contain many noisy features as shown in Figure 2.3(a). Although it is possible to manually crop an image to keep only the desired object (e.g. the church in Figure 3), it will be tedious to do so for every single database image. Besides, one may continually add new images to the database. Turcot and Lowe in [53] proposed to find useful features by matching an image against other database images containing the same object (i.e. the *friend* images) followed by geometric verification. Only top N_f features with most matched feature(s) are reserved for word quantization (as shown in Figure 2.3(b)). A word augmentation method is introduced so that an image can incorporate visual words from other images containing the same object, assuming that each image represents a different view of the same object. The authors also provided solutions to deal with a database of unlabeled images: each database image is used as a query image to find the top M matched images which are further verified using the geometric verification method, and the images with more than k matched features are regarded as *friend* images.

This solution has two problems:

Quantization Error: Matched features may be quantized into different visual words due to quantization errors[55]. Such "mismatches" cannot be detected and would degrade the retrieval performance.

Unbalanced Signature Length: Even though the authors suggested that each image reserves at most N_f matched features, the number of total matched features varies greatly[2] for each database image. Meanwhile, since multiple features can be

[2]In our experiment, the number of matched features per image ranges from 21 to 776 with a standard deviation of 89.

quantized to the same visual words, two images with the same number of features can have signatures with different numbers of visual words. This varying signature length has a negative impact on the retrieval performance which is confirmed by our experiment.

Simple Method for Balancing the Signature

We propose a simple improvement to the "Useful Feature" method to produce balanced image signatures of length N. It works as follows:

Step 1 *Balanced Feature Quantization*: Instead of reserving top N_f features, each image initially reverses all the features which have at least one matched feature in other *friend* images, in a ranked list (ranked by the number of matched features). Iterating though the ranked list, each accessed feature is quantized to a visual word to form the image signature. The iteration stops when the signature length reaches N or all the features in the ranked list have been accessed.

Step 2 *Balanced Word Augmentation*: After step 1, for those images with signature length smaller than N, we add words from their friend images[3] until the signature length reaches N. The friend images are ranked by the number of matched features, and we first add words from the higher ranked images.

Step 3 *Final Balance*: It is quite likely that after the first two steps, there are still images that have signatures shorter than N. For those images, we quantize the image features with no matched feature in "friend" images into visual words, and calculate the weighted value of those additional visual words using the *tf-idf* scheme. Those more heavily weighted visual words are incorporated into the image signature

[3]In our experiment, we also incorporate level-2 friends which are the friend images' friends as suggested in [53].

until the signature length reaches N.

This simple improvement solves that problem of unbalanced image signatures, but still has feature quantization errors.

Algorithm 1 Object Word Ranking Algorithm (Supervised Version).

Input: A set of raw image signatures $S_{raw} = \{Raw_1, Raw_2, ..., Raw_k\}$ for images containing the same object

Output: A set of compact signature $S_{cpt} = \{Cpt_1, Cpt_2, ..., Cpt_k\}$

```
1: S_cpt ← null
2: Ranked_List ← null
3: wordCountMap ← null
4: for each raw signature Raw_i ∈ S_raw do
5:     for each Word_j ∈ Raw_i do
6:         if Word_j ∉ wordCountMap then
7:             wordCountMap[Word_j] ← 1
8:         else
9:             wordCountMap[Word_j] + +
10: Sort wordCountMap by count
11: Ranked_List ← [word_i ∈ wordCountMap]
12: for each raw signature Raw_i ∈ S_raw do
13:     Cpt_i ← null
14:     for each Word_j ∈ Ranked_List do
15:         if Word_j ∈ Raw_i then
16:             Cpt_i.add(Word_j)
17:             if Cpt_i.length > N then
18:                 break
19:     S_cpt.add(Cpt_i)
    return S_cpt
```

Object Word Ranking

In this subsection, we introduce the *Object Word Ranking* algorithm which solves both problems of the "Useful Feature" method. Basically, for each object, we generate a ranked list of each word by its occurrence count in all images containing this object (i.e. the number of images containing this word). Thus, if a useful feature is quantized into different words with its matched feature in another image, those words will get a low rank in the ranking list. After that, each image ranks its own

words based on the word ranking list of that object, and the top N ranked words are reserved as the image signature. For words with equal occurrence frequency, we use their *tf-idf* weights to break the tie. For images containing two or more objects, we generate one signature for each object. The pseudo-code for the supervised scenario (i.e. all training images are labelled) is presented in Algorithm 1.

Algorithm 2 Object Word Ranking Algorithm (Unsupervised Version).

Input: The raw image signature of an image Raw_0 and the set of raw signatures $S_{raw} = \{Raw_1, Raw2, ..., Raw_k\}$ for its "friend" images
Output: The compact signature Cpt_0

$Wt \leftarrow 1$ ▷ Initial Weighting Factor
$Cpt_0 \leftarrow$ **null**
$Ranked\_List \leftarrow$ **null**
$wordCountMap \leftarrow$ **null**
for each $Word_j \in Raw_0$ **do**
    $wordCountMap[Word_j] \leftarrow W_0$
for each raw signature $Raw_i \in S_{raw}$ **do**
    $W_t \leftarrow W_t * p$
    for each $Word_j \in Raw_i$ **do**
        if $Word_j \in Raw_0$ **then**
            if $Word_j \notin wordCountMap$ **then**
                $wordCountMap[Word_j] \leftarrow W_t$
            else
                $wordCountMap[Word_j] += W_t$
Sort $wordCountMap$ by count
$Ranked\_List \leftarrow [word_i \in wordCountMap]$
for each $Word_j \in Ranked\_List$ **do**
    $Cpt_0.add(Word_j)$
    if $Cpt_0.length > N$ **then**
        break
return Cpt_0

For the unsupervised scenario where database images are not labeled, we adopt a similar method to that used in the "Useful feature" method to find "friend" images. Each "friend" image is ranked by the number of matched features, which are verified inliers using the RANSAC method. We further augment the "friend" list by appending each friend image's friends. Different from the algorithm for the

supervised scenario where the words existing in all images are treated equally, in the unsupervised scenario, each word existing in higher ranked "friends" is given a larger weighted count in the object word ranking, since we have a higher confidence that the higher ranked "friends" are true positives that do contain the same object. We use a scaling factor p $(0 < p < 1)$ to reduce the weight at each rank i:

$$W_i = W_0 * p^i \tag{2.1}$$

where W_0 is the initial weighting factor. We present the algorithm for the unsupervised scenario in Algorithm 2.

2.4.3 Top Inverted Index Ranking

Since the on-device mobile visual search system uses a relatively small dictionary, simply including all images sharing common word(s) with the query image as candidates will result in a large candidate set. Due to the limited screen size of mobile devices, we may present users with query results from those most-likely true-positive candidates, and let them submit additional queries if they desire more. Furthermore, users are only interested in the 1st ranked image for each query in mobile augmented reality applications. With the inverted index, we can return the top k images sharing the most common words as the query image using the *Top Inverted Index Ranking* method shown in Algorithm 3. Each query image then only compares against these top k candidate images, and hence makes real-time query possible on a small visual dictionary.

Algorithm 3 Top Inverted Index Ranking

Input: The signature of a query image Sig_{query} and the inverted index $index$
Output: A set of k candidate images $S_{img} = \{img_1, img_2, ..., img_k\}$

$S_{img} \leftarrow$ **null**
$Ranked\_List \leftarrow$ **null**
$imageCountMap \leftarrow$ **null**
for each raw word $Word_j \in Sig_{query}$ **do**
    $img\_List \leftarrow index[word_j]$
    for each $Image_i \in img\_List$ **do**
        if $Image_i \notin imageCountMap$ **then**
            $imageCountMap[Image_i] \leftarrow 1$
        else
            $imageCountMap[Image_i] + +$
Sort $imageCountMap$ by count
$Ranked\_List \leftarrow [image_i \in imageCountMap]$
for each $Image_i \in Ranked\_List$ **do**
    $S_{img}.add(Image_i)$
    if $S_{img}.length > k$ **then**
        break
return S_{img}

2.4.4 Re-ranking with Ranking Consistency

We use the ranking consistency method [57] to re-rank the returned list which significantly improves the retrieval performance. The ranking consistency scheme is based on the observation that two query images containing the same object would yield similar ranked lists. The ranking consistency scheme works as follows:

Initial Query: when querying an image in the database, an initial ranked list with top K images (selected from the k Candidate images) is returned based on certain similarity metrics such as the cosine distance;

Query Expansion: the top K images in the initial ranked list are used as query images to get their own ranked lists; a min-hash scheme is used for fast approximate retrieval;

Ranking-Consistency based Re-ranking: the top K images are re-ranked based on the similarity between each image's ranked list and the query image's ranked list. The re-ranked list L_{re} is built iteratively: at each iteration only one image is added to L_{re}, and each additional image is added based on its similarity to all the images already in the list. The similarity between two ranked lists l_i and l_j is measured using the *Rank Biased Overlap* (RBO) similarity which is calculated using Equation (2.2):

$$RBO(l_i, l_j, p, h) = (1-p)\sum_{d=1}^{h} p^{d-1}\frac{X_d}{d} \tag{2.2}$$

where p is a weighting factor similar to that defined in Equation (2.1) and a larger p is used for lower ranked results, h is the the number of results to compare, and X_d is the number of common words in the top d ranked results of both lists as shown in Equation (2.3):

$$X_d(l_i, l_j) = |l_i[1:d] \cap l_j[1:d]| \tag{2.3}$$

We modify the *Ranking Consistency* scheme using the two design principle discussed in Section 4.1.

Pre-calculated Ranked List for Database Images: When the image database is built, we pre-calculate the ranked list of each database image by using it as a query image. As discussed in the paper [57], for a dataset of V images, a list of top h ($h = [0.005^* V]$) results should be reserved for decent re-ranking performance. For a dataset of 10K images, 50 image IDs (2 bytes each) are stored to represent the ranking list. However, in our evaluation, we find that reserving just 25 image IDs

per image can achieve similar performance improvement. Therefore, we only use a pre-calculatd ranked list of 25 images which incurs an additional 50 bytes per image for lower query latency, and improved retrieval performance.

Fast Re-ranking with Direct RBO: Instead of calculating the re-ranked list iteratively, we simply calculate the RBO between the query image's ranked list and the ranked lists of all top K initially ranked images. The images are then re-ranked by the RBO to generate a new list to be returned to a user. The computing complexity is reduced from $O(K^2)$ to $O(K)$. As shown in our evaluation section, the iterative RBO method improves the retrieval performance slightly over this simplified direct RBO method, however, we can easily get a better retrieval result by re-ranking e.g. top *2K* results instead of K, which improves the retrieval performance significantly over the iterative method on the top K results, but still incurs a much smaller latency.

2.5 2-Stage KNN Re-ranking

In section 2.4.4, we use a k-NN based re-ranking method with ranking consistency measurement. However, there are two problems affecting its re-ranking performance. First, for efficiency consideration, only a fixed number of top K returned images are returned. However, if ground-truth images rank lower than K initially, such re-ranking algorithm cannot improve the ranking of those images. Second, ranking consistency measurement calculates the similarity of two ranking lists based on identical items which appeared in each list. However, a more meaningful ranking consistency measure should also consider the reciprocal k-NN relationship [67][68].

Two images are reciprocal k-NN if they are K nearest neighbors of each other, which is a strong indicator that two images are relevant. This means, for two ranked lists, we should not only consider the exactly matched images (i.e. common elements) but also images that are reciprocal k-NN. For example as shown in Figure 2.4, to calculate the k-NN similarity between n_1 and n_2, though image n_{1_3} (which is the same as n_t) and image n_{2_2} (which is the same as n_l) are not exactly the same image, they should still contribute to k-NN similarity since they are reciprocal k-NN.

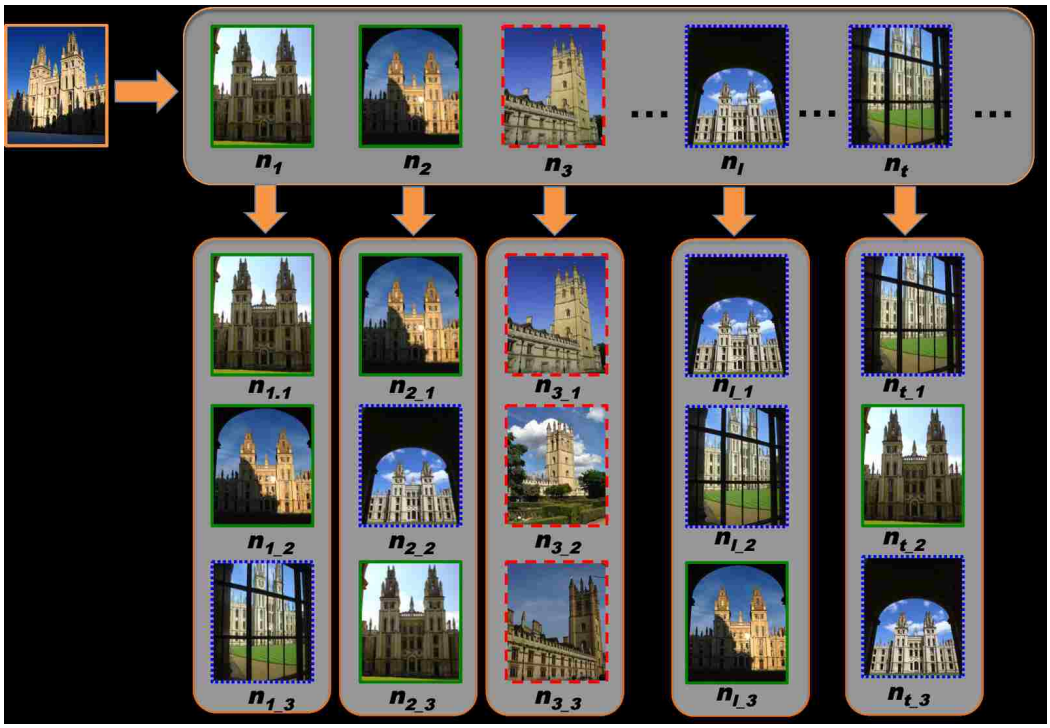


Figure 2.4: k-NN Re-ranking Example ($K = 3$). The rank of the 3rd nearest neighbor n_3, which is an irrelevant image (in red dashed box), will be dropped since the similarity between its k-NN to the query's k-NN is very low. On the other hand, the images ranked at positions l (n_l) and t (n_t) will be promoted since (1) they are reciprocal k-NN of the queries high-ranked neighbors (n_l is n_2's reciprocal k-NN and n_t is n_1's reciprocal k-NN), and (2) their k-NN lists are similar to the query's k-NN list (e.g. sharing common elements).

We propose a new two-stage k-NN re-ranking method. Different from [68][57] which only re-rank top K neighbors in the ranking list, in the first stage, we expand this list by exploring reciprocal k-NN relationship to incorporate potential relevant images (e.g. n_l and n_t in Figure 2.4). In the second stage, we re-rank this expanded list by giving each image a confidence score, which is iteratively calculated with a novel reciprocal k-NN aware ranking consistency measure.

2.5.1 *k*-NN Expansion

Instead of re-ranking a fixed number K of nearest neighbors, we expand the list by adding top neighbors' reciprocal k-NN which are potential relevant images that are low ranked (i.e. $> K$) in the preliminary ranked list. Here, we define $Top(q, k)$ as the set of k nearest neighbors for the image q, and then the set of reciprocal k nearest neighbors of image q is:

$$R(q, k) = \{p \in Top(q, k) \wedge q \in Top(p, k)\} \tag{2.4}$$

Specifically, we only add top $K^* = K/s$ ($s > 1$ and we use $s = 2$ for all experiments) neighbors' reciprocal neighbors to the expanded list to avoid adding too many noisy images. Meanwhile, with the help of the scaling factor s, we only need to maintain a relatively short list K^* of nearest neighbors for each database image, so that the memory cost for re-ranking purpose is significantly reduced. The procedure for k-NN Expansion is described in Algorithm 4.

Algorithm 4 k-NN Expanding

Input: $Top(q, K)$, the scaling factor s and threshold for reciprocal neighbor k
Output: Expanded nearest neighbor set S_{ext}

```
K* = K/s
S_ext = Top(q, K)
for each nb_i ∈ Top(q, K*) do
    for each nb_j ∈ R(nb_i, k) do
        if nb_j ∉ S_ext then
            S_ext.add(nb_j)
return S_ext
```

2.5.2 *k*-NN Re-ranking

With the expanded k-NN list, we calculate a confidence score for each of the image iteratively similar to that proposed in [57], and then re-rank the images in descending order of the confidence scores. To calculate the confidence score, we combine two similarity metrics to measure the similarity between two lists: the rank biased overlap (RBO) similarity and the reciprocal similarity.

The rank biased overlap (RBO) between two ranked lists is calculated using Equation 2.2. The reciprocal similarity between two ranked lists was recently proposed [68] to consider the reciprocal k-NN relationship between each pair of images from top l nearest neighbors of two query images i and j.

$$Recip(i, j, l, k) = \sum_{x \in Top(i,l)} \sum_{y \in Top(j,l)} f_r(x, y, k) \cdot w_r(i, x) \cdot w_r(j, y) \quad (2.5)$$

where $f_r(x, y, k)$ is a binary function indicating whether two images x and y are k reciprocal neighbors; w_r is a weighting function that gives higher weight to

higher ranked neighbors. However, the proposed linear weighting function w_r in [68] only works when each query image has the same number of ground-truth images (i.e. images representing the same physical objects with the query image). Here, we propose a new weighting function to remove this limitation:

$$w_r(i, x, r) = r^{\tau_i(x)} \tag{2.6}$$

where r is a weighting factor in the range $(0, 1)$, and the function $\tau_i(x)$ returns the rank of x in image i's ranked list.

We propose a new reciprocal k-NN aware RBO metric which integrates the reciprocal k-NN relationship into the ranking consistency measure. Specifically, we append the reciprocal similarity to the X_d factor in Equation 2.2 as shown in the following new equation:

$$rRBO(l_i, l_j, p, h) = (1 - p) \sum_{d=1}^{h} p^{d-1} \frac{X_d + Recip(i, j, d, k)}{d} \tag{2.7}$$

Now, we describe how the re-ranking works. After the expanded ranked list S_{ext} is generated in stage 1, we iteratively add images from this expanded list S_{ext} to a new re-ranked list, S, as follows: initially, we put s_1, which is the query image itself, into S. Next, we find s_2 which has the maximum similarity of ranked list as s_1 (measured by $rRBO$), and insert s_2 into S. Images that have been inserted into S will be removed from S_{ext}. Subsequently, for each of the rest ranking position k of the new list S ($k > 2$), we choose an image s_k from S_{ext} with the largest confidence score (Equation 2.9) and append this image to the end of S. This confidence score, $cs(x)$ is a product of the rRBO scores of this selected image with all previously

selected elements in S, i.e.

$$s_k = \underset{x \in \{x_i\}_{i=1}^{K} \setminus \{s_j\}_{j=1}^{k-1}}{\arg\max} cs(x) \tag{2.8}$$

$$cs(x) = \prod_{j=1}^{k-1} rRBO(x, s_j) \tag{2.9}$$

2.5.3 Re-ranking Speed-up

For fast re-ranking, we maintain a rRBO similarity table which captures the rRBO similarity score between a database image and its reciprocal nearest neighbors. To further improve performance, one can also use a fixed-size cache to store rRBO similarity scores of images from recent or popular queries.

2.6 System Evaluation

2.6.1 Datasets

We form our dataset by combining 12,753 landmark images of 4 datasets. The combined dataset contains 3658 ground truth images of 148 landmarks, and the number of groundtruth images per landmark ranges from 5 to 289. The rest of the 9095 images are used as distraction images which contain none of the 148 landmarks. To verify the performance of our system, a 5-fold cross validation is performed where each fold uses 1/5 of the 3658 ground truth images as query images ($\sim$ 732 images per fold). We choose to evaluate landmark recognition since the landmark images generally contain more noisy features which makes the recognition task more

challenging. Our system should works equally well on other image retrieval tasks.

ZuBud

The ZuBud [69] dataset contains images of 105 buildings in the Zurich city. Each building has 5 training images that are captured by two types of digital cameras at random view points. In addition, 115 query images are captured all of which contain buildings in the dataset.

Oxford and Paris

The Oxford Buildings dataset [70] and the Paris dataset [71] are landmark images collected from Flickr by searching for particular landmarks. The Oxford dataset contains 5062 images of 11 different buildings while the Paris dataset contains 6412 images of 12 different buildings. Each image is manually assigned to one of the following four labels: (1) *Good*: the building is clearly presented. (2)*OK*: more than 25% of the building is visible. (3)*Junk*: less than 25% of the building is presented. (4)*Absent*: the building is not visible. We only keep the *Good* and *OK* images as ground truth images, and hence we have 567 images from the Oxford dataset and 1790 images from the Paris dataset. The remaining 9095 images from these two datasets are used as distraction images.

LuBud

We have collected our Lehigh University Building dataset (LuBud) [72] using mobile devices (HTC Sensation 4G and LG G2). LuBud contains 181 images of 20 Lehigh University buildings.

2.6.2 System Performance Metrics

- **mAP** Mean average precision (mAP) is the average value of the average precision (AP) computed for each query image. The AP can be efficiently calculated using Equation 2.10:

$$AP = \frac{\sum_{k=1}^{n}(P(k) * rel(k))}{N} \tag{2.10}$$

where N is the number of ground truth images, k is the position in the ranked list, n is the position of the lowest ranked ground truth image, $P(k)$ is the precision of the top k returned results, and the function $rel(k)$ returns 1 if the item at rank k is a ground truth image, 0 otherwise [73]. AP is a value between 0 and 1, and 1 means perfect retrievals for a query image. The mAP metric is widely used to evaluate the image retrieval systems [49][57][52][53]. To calculate the mAP, the query image is compared against all database images to obtain a full ranked list.

- **ANMRR** For a mobile visual search system, users may care more about the quality of the top-ranked results instead of the complete ranked list. Therefore, we use the Averaged Normalized Modified Retrieval Rank (ANMRR) metric [74] designed by the MPEG group to evaluate our system. To calculate the ANMRR, a few definitions are needed :(a) **Rank**(k), the position in which the ground truth image k is retrieved; and (b) $K(q)$, the number of returned images. In a mobile visual search system, the number of returned images is generally limited by the screen size of mobile devices. In our experiment, we set $K(q)$ to 25. Therefore, if a ground truth image is not returned, its rank is

set to a large value as follows:

$$\mathbf{Rank}(k) = \begin{cases} \mathbf{Rank}(k) & : \mathbf{Rank}(k) \leq K(q) \\ 1.25 * K(q) & : \mathbf{Rank}(k) > K(q) \end{cases}$$

NG(q) denotes the number of ground truth images for a query image q,

$$\mathbf{NG}(q) = \begin{cases} \mathbf{NG}(q) & : \mathbf{NG}(q) \leq K(q) \\ K(q) & : \mathbf{NG}(q) > K(q) \end{cases}$$

Then, the average rank (AVR) of a query image can be calculated using Equation 2.11:

$$AVR(q) = \frac{1}{NG(q)} \sum_{i=1}^{NG(q)} \mathbf{Rank}(k) \tag{2.11}$$

The Modified Retrieval Rank (MRR) and Normalized Modified Retrieval Rank (NMRR) are calculated using Equation 2.12 and 2.13:

$$MRR(q) = AVR(q) - 0.5 * [1 + NG(q)] \tag{2.12}$$

$$NMRR(q) = \frac{MRR(q)}{1.25 * K(q) - 0.5 * [1 + NG(q)]} \tag{2.13}$$

NMRR is a value between 0 and 1, and 0 means a perfect retrieval for a query image q. Finally, the ANMRR is simply computed as the average of all NMRR values from all query images. ANMRR is also a popular metric to evaluate

Table 2.1: Mobile Devices in Experiment

Device	CPU	Memory	Android	Flash Storage	Release Date
HTC Sensation	1.2 GHz Dual-core	768 MB	4.0.3	1 GB	June 2011
LG G2	2.26 GHz Quad-core	2 GB	4.4.2	32 GB	Sep. 2013
Nexus 7 1st Generation	1.2 GHz Quad-core	1 GB	4.4.4	16 GB	July 2012
Nexus 7 2nd Generation	1.51 GHz Quad-Core	2 GB	4.4.4	16 GB	July 2013

image retrieval systems [75][76][77]. To calculate the ANMRR, a query is processed using the *Top Inverted Index Ranking* method to identify a list of the top K=100 database images with the most common visual words as the query image, and a ranked list of these top K=100 images are then generated.

- **Top1** Typicall, mobile users are only interested in the top ranked result e.g. the name of a building as in mobile augmented-reality applications. Using the first image in the ranked list generated from the ANMRR evaluation our Top1 image, we compute the recognition rate.

2.6.3 Experimental Setup

We implemented and evaluated our database construction pipeline on a Linux server with Intel(R) Xeon(R) CPU E3-1240 v3 @ 3.40GHz and 16 GB ram. For mobile efficiency, we use a few mobile devices released in the past 3 years including two smart phones: HTC Sensation and LG G2, and two tablets: Nexus 7 1st Generation and 2nd Generation. The characteristics of those devices are listed in Table 2.1.

For feature detection and descriptor extraction, we use the SURF64 implementation from OpenCV 2.4.9.

Table 2.2: The Performance of Various Dictionary Configurations.

Type	Size	Trees	mAP	ANMRR	Top1
Plain	5K	6 *kd*-trees	0.3289	0.7832	49.8%
Plain	10K	3 *kd*-trees	0.3428	0.6374	58.1%
Plain	15K	2 *kd*-trees	0.3531	0.5757	64.4%
Plain	**30K**	**1 *kd*-tree**	**0.3712**	**0.4928**	**69.2%**
VVT	20K	1 VVT	0.3424	0.5925	62.7%

2.6.4 Visual Dictionary and The Baseline

We compare the retrieval performance using different visual dictionary construction and feature quantization methods: the *Visual Vocabulary Tree* (VVT) and the *Randomized kd*-tree(s) on a plain dictionary. For our controlled experiment, we require that each method consumes about the same amount of memory for storing the visual dictionary, and about the same amount of CPU cycles on a single core for feature quantization. In our experiment, we use a B3D9 VVT (19683 visual words) as the benchmark for memory consumption and feature quantization latency, and then build different numbers of randomized *kd*-trees to compare against the benchmark. The Oxford 5K dataset [70] is used to train the visual dictionary, and the total number of extracted features is 4,475,961. For retrieval performance evaluation, all images retain all their visual words to generate the baseline performance result. Re-ranking is not performed in this experiment.

Our experimental results are presented in Table 2.2[4]. We find that a plain dictionary built with randomized *kd*-trees still outperforms VVT even with fewer visual words (Plain 15K vs. VVT 20K). On the other hand, a larger visual dictionary with fewer randomized *kd*-trees produces much better performance than a smaller

[4]Re-ranking methods such as geometric verification or ranking consistency are not applied.

visual dictionary with more randomized *kd*-trees, meaning that reducing the number of trees only has a small impact on the approximation accuracy for the nearest neighbor search. Therefore, we conclude that with constrained memory size and CPU cycles, the plain dictionary with a single *kd*-tree method should be adopted for constructing our visual dictionary.

All remaining experiments reported subsequently use a plain 30K dictionary on a single *kd*-tree, and the performance result highlighted in Table 2.2 is used as the baseline performance.

2.6.5 Compact Signature

We compare the following methods for generating compact signatures:

TopSurf: The authors in [23] suggested that only the top N most frequent words are retained as the image signature. However, since many words may have the same frequency, we further rank those words using their *tf-idf* weights.

Useful Feature: This uses the method introduced in [53] which retains at most N features per image with word augmentation.

Balanced Signature: We apply the simple improvement discussed in section 5.2.2 to the *Useful Feature* method that each database image retains N visual words.

Word Object Ranking: Each image retains N words using the *Word Object Ranking* algorithm proposed in section 5.2.3.

For the last 3 methods, we evaluate both the supervised and unsupervised scenarios. In the unsupervised scenario, to decide whether two images contain the same object, the threshold for the minimum number of matched features (with RANSAC verification) is set to 20. For images with no matched images, we use *TopSurf* to

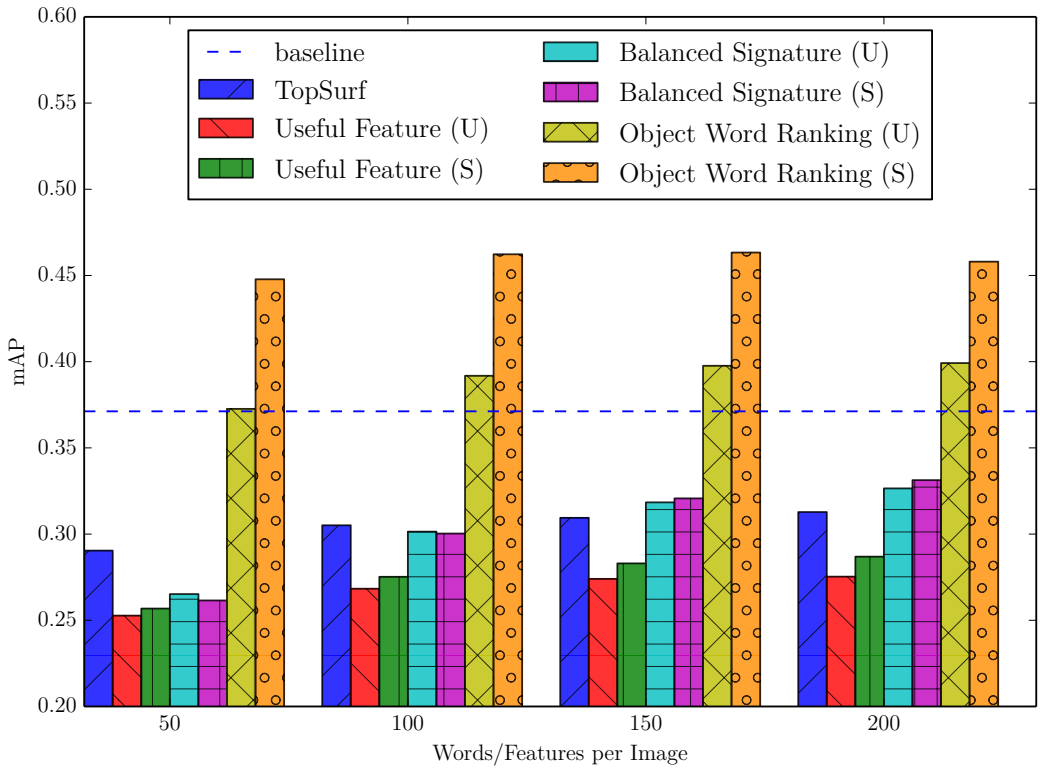


Figure 2.5: mAP. (S) denotes the supervised scenario and (U) denotes the unsupervised scenario.

generate their image signature.

Retrieval Performance

The retrieval performance results are plotted in Figure 2.5, Figure 2.6 and Figure 2.7. Our proposed *Object Word Ranking* (OWR) algorithm is the only method with obvious improvement over the baseline system without compact signature. Furthermore, our OWR method outperforms other methods. The other methods generally perform worse than the baseline system when a small number of words/features are used or similar to the baseline system when a large number of words/features are

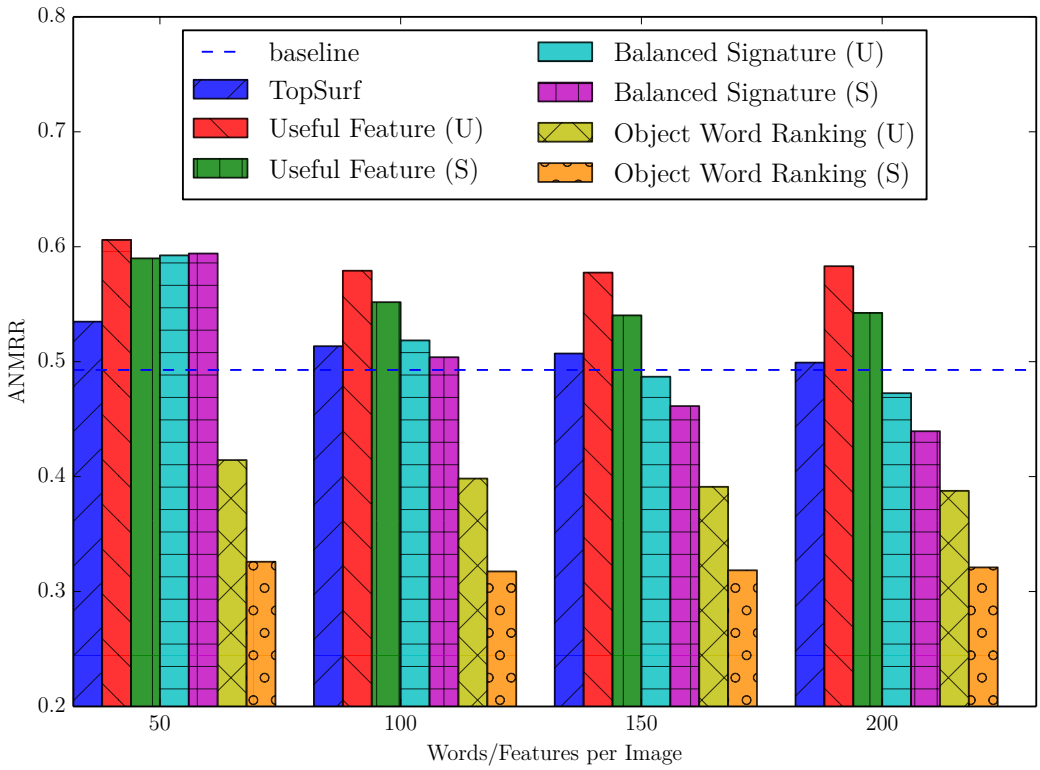


Figure 2.6: ANMRR. (S) denotes the supervised scenario and (U) denotes the unsupervised scenario.

used. The native "Useful Feature" method performs even worse than the simple "TopSurf" method while the balanced signature length method outperforms "TopSurf" when their signature length exceeds 150 words.

In the supervised scenario, OWR achieves the best mAP score of 0.4633 for using the top 150 ranked words as the image signature, but it also has a good score of 0.4623 when using only the top 100 words, which is almost 25% improvement over the baseline system. OWR achieves the best ANMRR score of 0.3175 when the top 100 words are used, which is over 35% improvement to the baseline system. The Top1 recognition rate of OWR is above 73.5% when more than 100 visual words are

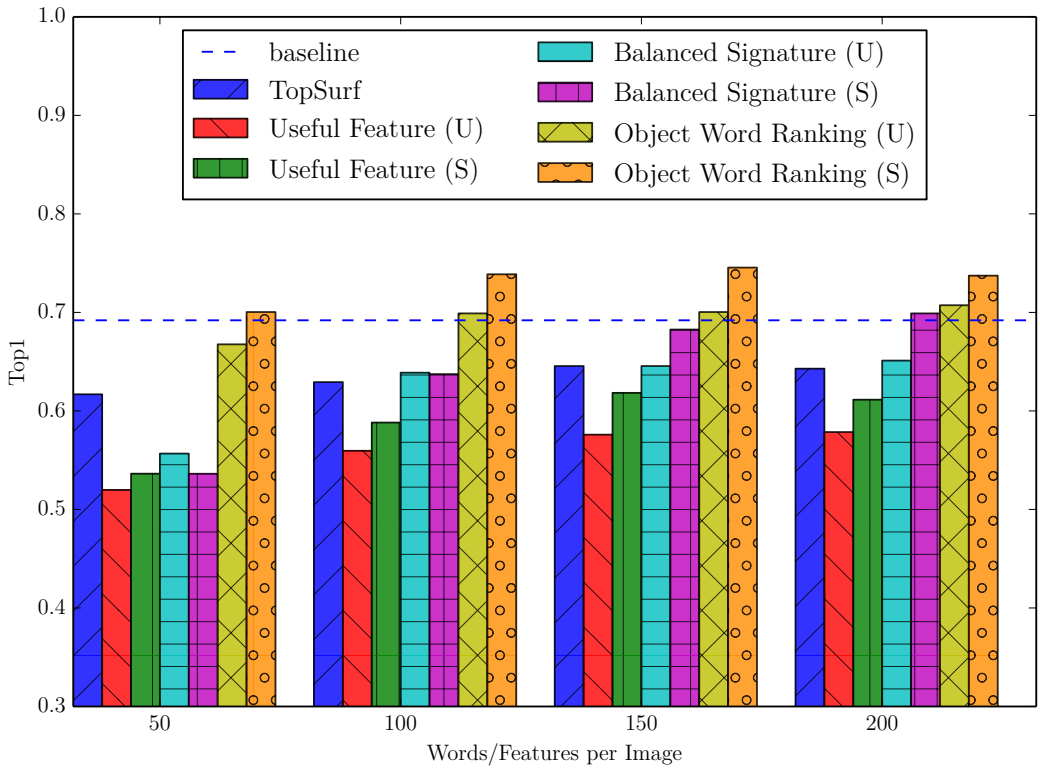


Figure 2.7: Top1. (S) denotes the supervised scenario and (U) denotes the unsupervised scenario.

used, and this introduces a performance improvement of 7%.

In the unsupervised scenario, the performance of OWR increases monotonically with the number of reserved visual words. However, the performance of using 100 visual words is only slightly worse than that achieved using 200 visual words.

Memory Reduction

Using the 30K visual dictionary, on the average, each image has a raw signature of 651 words. With the compact signature generated using the *Object Word Ranking* algorithm (100 words per image), we are able to reduce the memory cost per image

by 85%.

2.6.6 Ranking Consistency based Re-ranking

We evaluate the retrieval performance using the *Ranking Consistency* re-ranking method (section 2.4.4). For this experiment, we represent each image with 100 visual words generated by the Object Word Ranking algorithm in the supervised scenario. We set the weighting factor p to 0.9 as used in [57].

Iterative RBO vs Direct RBO

Table 2.3 (K denotes the number of images to re-rank) compares the two re-ranking methods by applying ROB on the top 25 ranked items on each image's ranked list. As shown in the table, the iterative scheme achieves slightly better performance over the direct scheme but introduces significantly more re-ranking latency[5]. However, a better performance can be achieved for the direct scheme when more images are re-ranked (e.g. 50-100), which only increases minimal re-ranking latency (3 ms for re-ranking 50 more images). On the other hand, the latency for applying the iterative method increases significantly if more images needs to be re-ranked, i.e. 270 ms for re-ranking 50 more images.

Performance Improvement

We evaluate the performance improvement by applying the *Ranking Consistency* re-ranking scheme. As suggested in the paper [57], for a dataset with V images,

[5]The time here is measured on the server-side, and an even higher latency can be observed on a mobile device.

Table 2.3: Iterative RBO vs Direct RBO

Method	***K***	**mAP**	**ANMRR**	**Top1**	**Time**
Iterative	50	0.4941	0.2770	74.42%	200 ms
Direct	50	0.4907	0.2771	74.42%	2 ms
Iterative	100	0.5059	0.2718	74.42%	470 ms
Direct	100	0.4989	0.2723	74.42%	5 ms

each database image should have a ranked list of top h ($h = [0.005 * V]$) matching images. Therefore, for a 10K dataset, each image is supposed to maintain a list of 50 image IDs (100 bytes). In this experiment, we also evaluate the re-ranking performance for the scenario where each image only maintains a ranked list with 25 image IDs (50 bytes).

The experiment results are presented in Table 2.4. In the table, K denotes the number of images to re-rank, and h denotes the number of image IDs in the ranked list of each image. The mAP and ANMRR scores increase with increasing numbers of re-ranked images. On the other hand, only slight improvement is observed for the Top1 retrieval result. Meanwhile, we find that reducing the number of image IDs by 50% has little impact on the re-ranking performance. We get an even better *Top1* recognition result on a smaller h value.

2.6.7 Total Memory Cost For a 10K Database

Based on our presented experiment results, we can build an efficient on-device mobile visual search system with a relatively small 30K visual dictionary ($\sim$ 7.5 MB). For a decent retrieval performance, each image can be represented with a compact signature ($\sim$ 300 Bytes) consisting of 100 visual words and their associated word

Table 2.4: Performance Improvement with Ranking Consistency

K	h	mAP	ANMRR	Top1
25	25	4.52%	5.32%	0.81%
25	50	4.56%	5.61%	0.56%
50	25	6.14%	12.7%	0.81%
50	50	6.25%	13.57%	0.56%
75	25	7.12%	14.0%	0.81%
75	50	7.33%	14.96%	0.56%
100	25	7.92%	14.2%	0.81%
100	50	8.22%	15.18%	0.56%

frequencies. An additional 50 Bytes per image is used to store a ranked list of 20 image identifiers for our ranking consistency based re-ranking method. Meanwhile, each database image consumes 200 Bytes in the inverted index table for fast query purpose. The IDF weighting vector on the 30K visual dictionary consumes about 117 KBytes memory. The total memory cost is less than 13 MB.

2.6.8 Mobile Efficiency

We implemented our prototype system as well as an augmented reality application on the Android platform. For mobile efficiency test, we randomly select 5 images from

Table 2.5: Query Images

Landmark	Feature No.	Words No.
Business Center	930	874
Engineering Library	953	906
Linderman Library	417	399
Christmas-Saucon Hall	442	427
Packard Lab	577	543
Average	**663.8**	**629.8**

LuBud as our query images, and build an image database using the remaining 12748 images from the combined dataset. Each database image is represented with the 100 most useful words selected using our supervised Object Word Ranking algorithm. The number of features and quantized words for each query image are summarized in Table 2.5.

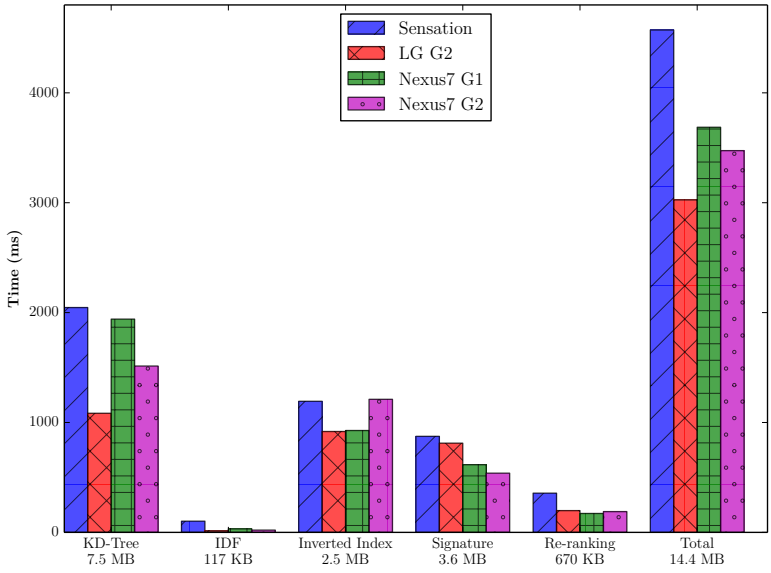


Figure 2.8: Image Database Loading Time

Database Loading Time

The start-up time is critical for a mobile visual search system, for a ODMVS needs to load the image database into system memory during initialization. For fast database loading, we use the Kryo serialization framework [78] to store the image database as binary files which can be efficiently loaded into the system memory.

Figure 2.8 plots the loading time for the image database as well as the supporting data structures on different mobile devices. Generally, all devices have a loading

time of less than 4.5 seconds making the system immediately available after bootup. Loading the visual dictionary (i.e. the *kd*-tree) is the most time consuming part due to its size, and the loading efficiency depends on the CPU frequency of each device. The old *Sensation* device has a small internal flash storage, so we store the binary files on the external SD card. Therefore, their file loading processes are much slower than other devices which load the files from the internal storage directly. On the other hand, the Nexus Tablets are very fast on loading files from the internal flash storage.

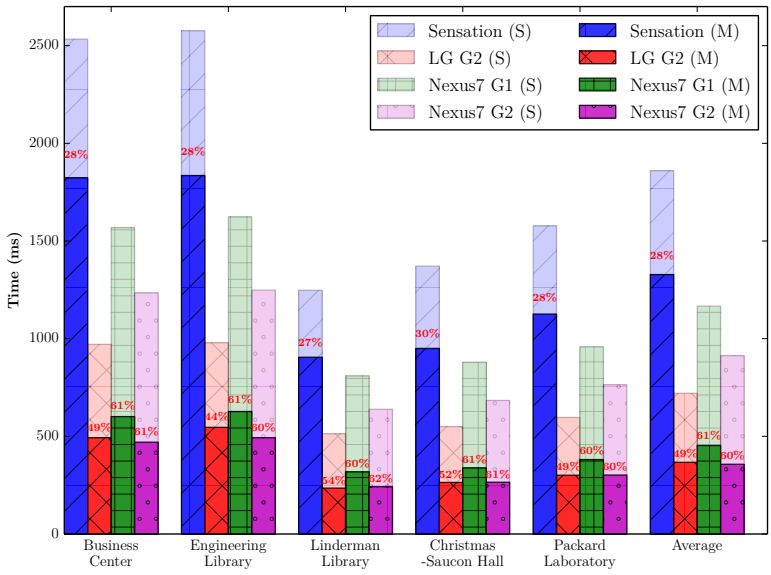


Figure 2.9: Total Query Latency: (S) denotes result of single threaded computing model; (M) denotes result of multi-threaded computing model. The latency reduction is given with percentage in red color.

Query Latency

We measured the query latency of five query images using the 4 Android devices listed in Table 2.1. Both single-threaded and multi-threaded computing models are used, and our experiment results are shown Figure 2.9.

Single-threaded computing model: For the single-thre-aded computing model, the query latency is largely determined by the CPU frequency. On the average, query responses are returned within 1-1.5 seconds for those devices with quad-core CPUs. For the HTC Sensation with only 2 cores, it takes much longer time (more than 2 seconds) than the device with the same CPU frequency but having more cores (i.e. the 1st Generation Nexus 7). One possible reason is that many background processes, such as phone and message service, which normally have higher priorities than user applications, run on that HTC Sensation phone.

Multi-threaded computing model: The multi-threaded computing model can be applied to those computing components with repeated computing tasks to speed up such computations. Thus, we convert several more time-consuming components, e.g. feature descriptor extraction, feature quantization, top inverted index ranking, and similarity calculation, in the query processing pipeline into multi-threaded computing tasks. As shown in Figure 2.9, a query submitted using newer quad-core devices can receive a response in about 500 ms ($\sim$ 50% improvement), and about 1.7 seconds (28% improvement) on the old dual-core device.

Concurrency Analysis: Modern mobile devices are equipped with multi-core CPUs. To speed up the query process, we make the maximum parallelism by exploiting the benefit of multi-threaded computing model. The multi-threaded computing model can be applied to those computing components with repeated

computing tasks on a large number of inputs, and thus the following four most time-consuming components in the query processing pipeline can be converted into the multi-threaded computing tasks.

(1) *Feature detection and description*: the image features can be efficiently detected with fast Hessian detector that is introduced in [21], however, the descriptor calculation on those detected features is a time consuming task. Therefore, we calculate the feature descriptors concurrently on each CPU core after the features are detected, i.e. each core work on one partition of the image. The multi-threaded computing model largely reduce the feature description latency for almost 50% on all devices.

(2) *Feature Quantization*: we partition the feature descriptor list into several segments and send each segment to a single thread, so multiple features can be quantized to visual words concurrently on each CPU core. With 4 CPU cores, upto 75% reduction on the computing time can be achieved.

(3) *Top Inverted Index Ranking*: for each visual word in the query image, we get the list of database images containing this word and update their counts in a Hash Table. However, since the operation of reading the image list is very quick, but the Hash Table updates must be performed in a synchronized way, the multi-threaded computing model actually slow down this process. Therefore, we keep using the single-threaded computing model on this query component.

(4) *Similarity Calculation*: we partition the candidates image list into several segments and send each segment to a single thread, so multiple candidates images can be compared concurrently on each CPU core. There isn't much improvement when more CPU cores are involved in the processing, and the devices with higher

CPU frequency show smaller latency reduction, e.g. on average LG G2 only has 17% reduction, but the 1st generation Nexus7 has 35% reduction. The reason is that the calculation of cosine distance between two image signatures is relatively fast, and in total we only have to compare 100 candidates images generated using the *Top Inverted Index Ranking* method. Therefore, the benefit of using parallelism is compromised by the cost of data preparation and thread management.

Discussion

Some query processing components can be further optimized to reduce query latency: (1) restricting the number of features (e.g. 200 features as used in [49]) extracted from a query image would decrease the latency for feature extraction, feature quantization, and the *Top Inverted Index Ranking*; (2) reducing the number of candidate images (e.g. selecting k=50 candidates instead of 100) would reduce the similarity calculation and re-ranking latency. However, such approaches may degrade the retrieval performance for certain application scenarios and we intend to investigate these issues in our future work.

2.6.9 Evaluation for 2-Stage k-NN Re-Ranking

We first present presents three sample re-ranking results using our proposed method in Figure 2.10. They show that our method returned more similar images in the top-ranked list. In the rest of the subsection, we show that large retrieval accuracy improvement can be achieved without any obvious sensitivity to the settings of tunable parameters. In addition, we also demonstrate that our proposed method outperforms other state-of-the-art k-NN based re-ranking approaches.

Figure 2.10: Three examples of re-ranking from Holidays dataset. For each query (left), initial ranked list (the first row) and re-ranked list (the second row) are demonstrated.

Dataset	Initial Result		Re-ranked Result	
	Baseline BOVW	Advanced BOVW	Baseline BOVW	Advanced BOVW
UKbench	74.58%	85.90%	81.14%	91.86%
Holiday	53.58%	71.88%	57.19%	77.21%
Holiday+1 million	N/A	51.79%	N/A	57.90%

Table 2.6: mAP for both inital and re-ranked retrieval results

Experimental Setup

Datasets: we use two popular benchmark datasets, namely (i) the UKbench dataset[6] and (ii) the Holiday dataset[7]. We also add one million distraction images [8] to the

[6] `http://www.vis.uky.edu/~stewe/ukbench`
[7] `http://lear.inrialpes.fr/~jegou/data.php`

Holiday dataset for large scale evaluation.

Performance Metric: We use the typical *mean average precision* (**mAP**) as our retrieval accuracy metric.

Initial Ranked List: For each benchmark dataset, we generate two initial ranked lists: one low accuracy list using the baseline *bag-of-visual-word* (BOVW) model; and one high accuracy list using the advanced BOVW model with several recently proposed enhancement schemes such as *multiple assignment* (MA) [79], *hemming embedding* (HE) [80] and *burstiness management* (BM) [81]. Both models are constructed using the SIFT local features. For the large scale experiment, we only generate one initial ranked list with the advanced model. Table 2.6 presents **mAP** for both initial ranked list and our re-ranked list.

Effectiveness of Stage 1: The Expanded Candidate Set

We want to evaluate the performance of the expanding strategy in stage 1 of our algorithm, i.e. whether it can find the initially low ranked ground truth images without adding too many noisy images to the candidate set that will be re-ranked in stage 2. We run our experiment on the UKbench dataset that is generated using the baseline BOVW framework.

First, we show in Figure 2.11 that the total number of missed ground truth images when different number of top ranked images (K in **Algorithm 4**) are selected as candidate images for re-ranking. It is clear that a high threshold will reduce the number of missed ground truth images. However, the slope rate of this graph is decreasing rapidly, meaning that it is not a wise idea to simply set a higher threshold

[8] `http://press.liacs.nl/mirflickr/`

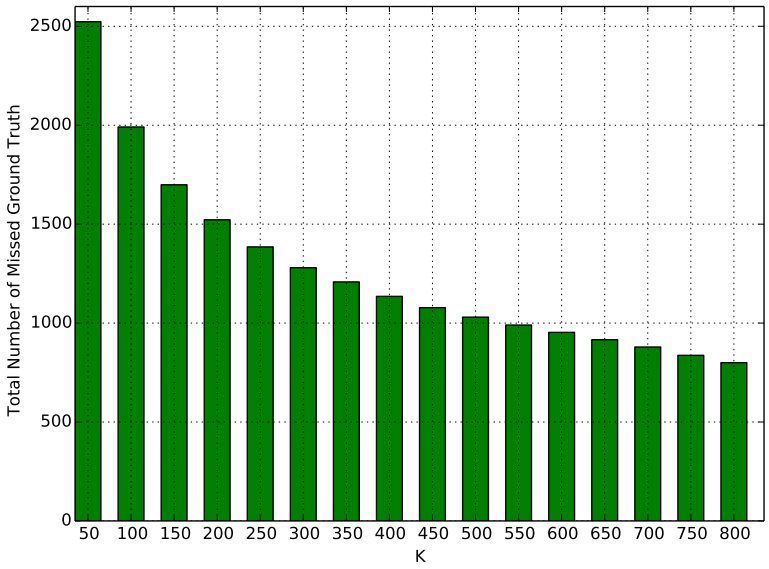


Figure 2.11: The total number of ground truth images that are excluded for re-ranking when different number of top ranked images K are used for UKbench dataset.

to increase the recall, considering that the larger number of candidate images means greater re-ranking latency. For example, when a threshold of 200 is used, around 1500 ground truth images are missed for re-ranking; when the threshold is increased to 400, there are still almost 1200 ground truth images are not included.

Next, we run our candidate expanding experiment by fixing the length of candidate images (K in **Algorithm 4**) before expansion to the top **50** ranked database images of each query image. We experiment by varying the threshold of reciprocal neighbor k in Equation 2.5. Figure 2.12 shows the total number of ground-truth images that are added to the expanded candidate image set for re-ranking, and Figure 2.13 shows the average number of more candidate images that are added to the

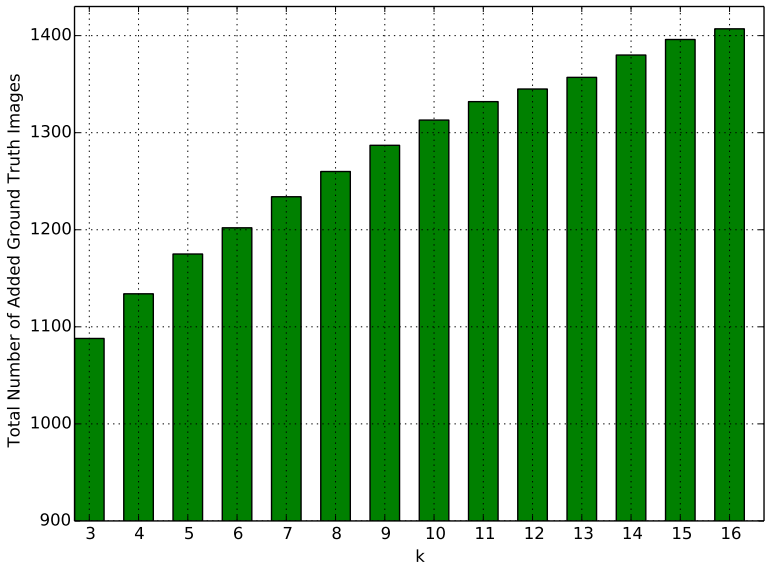


Figure 2.12: The total number of low-rank ground truth images that are promoted for re-ranking after expansion under different k for UKbench dataset (K=50).

candidate set after expansion. The result demonstrates that our candidate set expanding method is quite effective that we incorporate a large portion of low-ranked ground truth images without adding too many noisy images. For example, when k is set to 10, more than 1300 of the total 2500 missed ground truth images (52%) are added to the expanded list while on average we only need to re-rank 37 more images for each query (i.e. 87 in total). To achieve the same amount of recall, the method using fixed length of candidate image set has to re-rank more than 350 top ranked candidate images per query which is about 4 times of our strategy (87 per query).

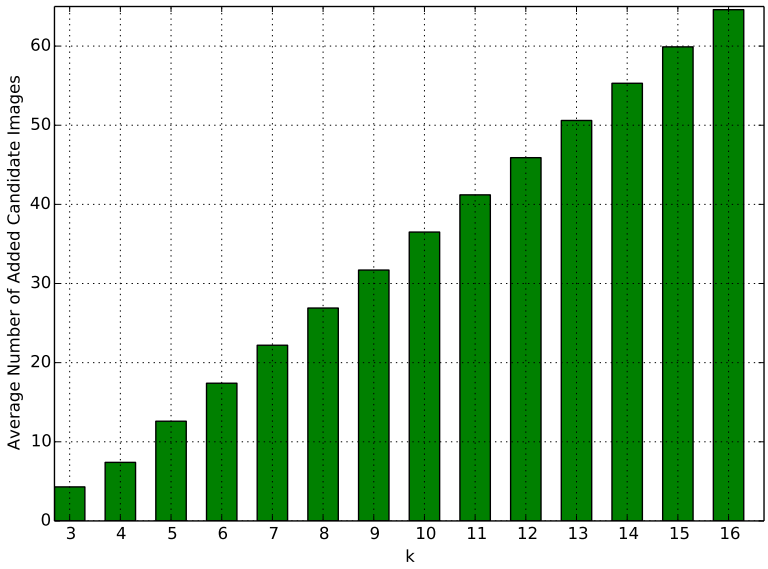


Figure 2.13: The average number of promoted low-rank images (both ground truth and noise) per query under different k for UKbench dataset (K=50).

Parameter Setting

Here we want to demonstrate that our re-ranking method is not quite sensitive for parameter selections of k (reciprocal neighbor size) and r in Equation 2.5, which is a severe issue for some previous works [67][68]. We set the length of the initial ranked list to 50. We also set h to 25 and p to 0.6 in Equation 2.7, as in [57]. Using the Holiday dataset for our experiment, we try different values of k (reciprocal neighbor size) and r in Equation 2.5 to find suitable ones for optimal performance improvement.

First, we fix k as 10 and evaluate the re-ranking result using different scaling factor r. As shown in Table 2.7, good performance improvement can be achieved when r is in the range of $[0.3, 0.6]$, and we find that the optimal value of r is around 0.4.

r	Baseline BOVW	Advanced BOVW
0.1	56.81%	76.73%
0.2	56.82%	76.67%
0.3	57.34%	76.94%
0.4	**57.65%**	77.23%
0.5	57.54%	77.20%
0.6	57.28%	**77.26%**
0.7	57.11%	76.70%
0.8	56.97%	76.28%
0.9	55.85%	76.12%

Table 2.7: Re-ranking by tuning r ($k = 10$)

k	Baseline BOVW	Advanced BOVW
5	57.11%	76.13%
6	57.01%	76.25%
7	56.90%	76.91%
8	57.19%	77.21%
9	57.23%	**77.51%**
10	**57.65%**	77.23%
11	57.32%	77.12%
12	57.70%	77.25%
13	57.30%	77.27%
14	57.56%	77.26%
15	57.59%	77.26%

Table 2.8: Re-ranking by tuning k ($r = 0.4$)

Next, we fix r as 0.4 and investigate how different values of k affect the re-ranking performance. The result is shown in Table 2.8. We observe that the largest performance improvement was achieved when k is in the range of $[8, 12]$.

We evaluate the re-ranking results over the UKbench dataset and the large-scale dataset with $r = 0.4$ and $k = 8$. For the UKbench dataset, the mAP improved from

Method	Baseline BOVW	Advanced BOVW
Reciprocal k-NN (1) [82]	56.81%	75.14%
Reciprocal k-NN (2) [68]	53.34%	72.13%
Hello Neighbor [67]	56.93%	75.01%
Ranking Consistency [57]	56.17%	75.27%
Ours ($k = 8, r = 0.4$)	**57.19%**	**77.21%**

Table 2.9: Re-ranking comparison with other methods (Holiday)

Method	Baseline BOVW	Advanced BOVW
Reciprocal k-NN (1) [82]	80.32%	90.02%
Reciprocal k-NN (2) [68]	76.01%	89.62%
Hello Neighbor [67]	78.92%	89.02%
Ranking Consistency [57]	79.35%	90.14%
Ours ($k = 8, r = 0.4$)	**81.14%**	**91.86%**

Table 2.10: Re-ranking comparison with other methods (Ukbench)

74.58% to **81.14%** using the baseline BOVW model and from 85.90% to **91.86%** using the advanced BOVW model. For the large-scale experiment, the mAP improved from 51.79% to **57.90%**.

Comparison to other *k*-NN Re-ranking methods

We compare our method with some of the state-of-the-art k-NN based re-ranking solutions, and we name those methods as *Reciprocal k-NN (1)* [82], *Reciprocal k-NN (2)* [68], *Hello Neighbor* [67], and *Ranking Consistency* [57] respectively. For each method, we tune the parameters to achieve maximum performance improvement. For a summary of these methods and their optimal parameter selection, please refer to our extended report `http://www.cse.lehigh.edu/~dal312/knn_rerank.pdf`. As

Method	Advanced BOVW
Reciprocal k-NN (1) [82]	56.85%
Reciprocal k-NN (2) [68]	55.36%
Hello Neighbor [67]	54.17%
Ranking Consistency [57]	55.58%
Ours ($k = 8, r = 0.4$)	**57.90%**

Table 2.11: Re-ranking comparison with other methods (1 Million)

shown in Table 2.9 and Table 2.10, our 2-stage k-NN method achieves the best results under all experimental scenarios.

2.7 Summary

In this chapter, we have presented the design and evaluation of EMOD, an efficient on-device mobile visual search system. EMOD is based on the Bag-of-visual-words framework and we propose several optimization techniques to reduce its memory cost and the query latency. Compared with the baseline system, EMOD achieves up to 85% memory reduction on image signatures and provides significantly improved retrieval performance. Real-time image query is guaranteed by optimizing each component in the query processing pipeline. In addition, we also presented the proposed 2-stage k-NN re-ranking method and showed that it performs better than state-of-the-art in terms of efficiency and accuracy. We implemented a prototype system on Android platform and demonstrated its effectiveness using an augmented reality application for landmark recognition.

Chapter 3

Moca: Energy-Efficient and Privacy-Preserving Deep Processing for Mobile Vision Applications

In this chapter, we present Moca, an energy-efficient and privacy-preserving deep processing system for vision-based applications. The key idea of Moca is to decouple the hierarchical neural network layers and splits them across mobile and cloud for energy-efficiency, and leverage the privacy-preserving feature maps extracted from deep models to protect personal data privacy. Moca also enables differentiated privacy settings via a DNN-based privacy quantification framework to meet the varying privacy requirements of different mobile applications. Our experiments with various mobile vision applications against thousands of photos and videos from a variety

of validation datasets demonstrate that MOCA has up to 6.7x lower (85% less) energy overhead on mobile devices compared to the state-of-the-art approaches, while providing stronger data privacy with negligible impact on application accuracy.

3.1 Introduction

With the development of more advanced machine learning technologies recently, researchers have rushed to apply deep neural network (DNN) techniques in mobile applications [83–88]. Among these applications, an important category that benefits from DNNs is mobile vision based application. Typical examples include object, scene, face recognition and detection. For instance, a mobile application could use DNNs to recognize the objects that appeared in the photos and videos taken by mobile cameras [89]; a mobile payment application could leverage DNNs for face recognition to verify the user's identity [90]; an augmented reality application running on mobile devices uses DNNs for object and scene detection and recognition before augmenting interesting information to users [91, 92] and so on.

However, running DNNs on mobile device faces performance and energy challenges because of its high requirements on memory, computation and battery resources [88, 93], which cannot be satisfied by commodity mobile devices today. For example, processing a 5 minutes video with a highly accurate object recognition model on a standard mobile device consumes 53% of its battery capacity (see Figure 3.1). Leveraging hardware technologies such as GPU and hardware accelerators could accelerate DNN performance, however, they are often not available on commodity mobile devices [93]. Furthermore, mobile devices are still constrained by

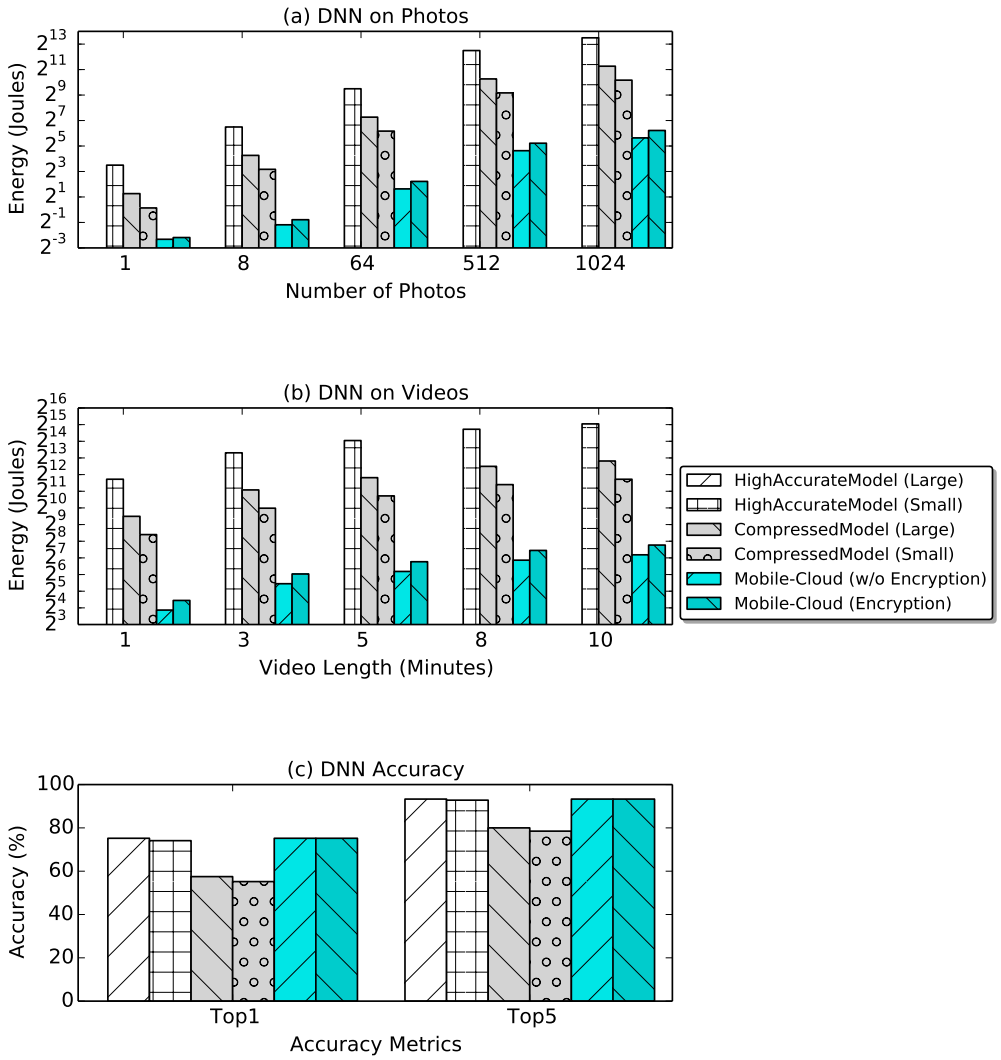


Figure 3.1: Motivation scenarios for MOCA: deep processing photos and videos with object recognition DNNs on mobile device directly vs. offloading data to cloud via WiFi for further deep processing with highly accurate DNN models. Compared to highly accurate models, compressed models consume 5–10x (80%–90%) less energy as shown in (a) and (b) at the cost of dropping DNN accuracy by 18.3% as shown in (c); the mobile-cloud offloading approaches consume 155–232x (99.35%–99.57%) less energy but run the risk of leaking personal data on untrusted cloud. The large highly accurate model fails in all application cases due to the limited memory capacity on modern mobile devices.

their size, weight and battery capacity [94]. Software-based optimizations such as model compression [88] were proposed recently to reduce the memory footprint and computation cycles during DNN execution, however they compromise the model accuracy and still consume significant battery power.

Deploying DNN services on cloud can shift the computational burden from mobile devices to powerful shared servers in the cloud. However, we have to address the privacy challenge on the untrusted cloud [95, 96]. Encrypting users' personal data such as photos and videos on mobile devices not only introduces additional energy overhead, but also complicates the data management for mobile applications [97, 98]. To make matters worse, information from such encrypted data could still be leaked to the untrusted cloud during the deep learning process [99–101].

In this chapter, we present MOCA, which uploads only privacy-preserving visual features extracted from users' data to the cloud, while utilizing the cloud for high-performance deep processing to produce very high accuracy models for mobile vision applications. The key idea of MOCA is inspired by the observation that the extracted features from the layers of a deep neural network have undergone combined linear and non-linear data transformations, and hence such features provide some form of privacy to the users. Features extracted from deeper layers provide higher privacy (see Figure 3.3). Without the original DNN model, adversaries cannot infer additional information regarding the extracted features even if they use a highly accurate DNN-based object recognition and detection system. Moreover, compared to the data encryption approach for achieving privacy, MOCA achieves lower performance and energy overheads since it only requires executing feature extraction on mobile devices. To further accelerate the feature extraction procedure

and reduce the feature size, several optimizations including layer decomposition and parameter-free layer merging are proposed and implemented in Moca.

Moca enables differentiated privacy for different mobile applications. For instance, a mobile payment application could require higher privacy than a common mobile object recognition application. To evaluate the privacy level of features extracted from a deep neural network layer, a DNN-based privacy quantification framework is proposed to enable application developers to calculate a reference privacy value based on their own defined criteria. The feedback from these evaluations will guide the adaptive DNN model reconstruction process to determine the number of layers executed on the mobile device and optimize those layers.

To summarize, we make the following contributions:

- We present a deep processing system named Moca for mobile vision applications. It decouples the hierarchical layers of DNN models and splits them across mobile and cloud. It uses privacy-preserving feature maps extracted from DNNs on mobile as the input for further deep processing on cloud to protect personal user data.

- We propose five novel techniques for improving the energy efficiency of the feature map extraction process on mobiles which include layer decomposition, parameter-free layer merging and branch reduction .

- We propose a DNN-based framework that allows mobile application developers to quantify the privacy level for user data based on their own privacy criteria.

- We present the differentiated privacy mechanism for DNN-based mobile applications, and enable adaptive DNN model reconstruction to efficiently utilize

mobile resources and achieve better privacy guarantee.

We design and implement MOCA based on the open source deep learning framework Caffe. MOCA has two major components implemented distributedly across mobiles (for feature extraction) and cloud (for deep processing and privacy evaluation). Experimental results with a variety of DNN-based mobile applications on several large-scale datasets demonstrate that the privacy of these data are well preserved, while the energy overhead on mobile device is 1.2x - 6.7x (16.7% - 85.1%) lower than the state-of-the-art approach.

The rest of this chapter is organized as follows: § 3.2 presents the challenges that we address in this work. Design and implementation of MOCA are described in § 3.4. Evaluation results are shown in § 3.5. § 3.6 presents the related work. We conclude the chapter in § 3.7.

3.2 DNN Challenges on Mobile

Mobile deep processing presents new challenges for mobile system design. It empowers intelligent services for mobile applications, however, the limited battery, memory and computation resources on mobiles force system designers to run deep processing on the remote cloud, making the privacy issue a serious concern for such mobile applications.

DNN tasks are computationally intensive: It is well known that deep neural network tasks often require much memory and computation resources. For example, a large and highly accurate DNN model with 152 forward and backward propagation layers has 230 MB parameters [31]. It requires 1.7 GB memory during

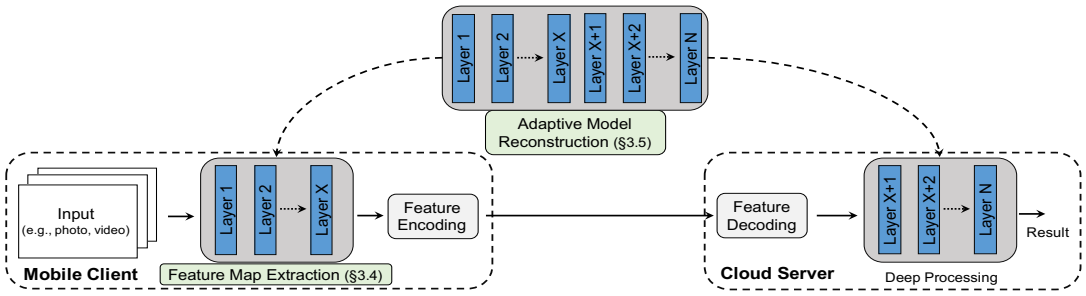


Figure 3.2: System overview of MOCA. It uses the feature maps extracted from the first few layers in DNN models as the input for further deep processing on cloud. These extracted feature maps preserve strong privacy for user data and high accuracy for application services. Moreover, MOCA has low energy overhead, as only a few layers of DNN models are executed on mobiles, and several optimizations on model reconstruction and feature extraction are proposed for improving its energy efficiency.

the runtime and cannot be executed on a standard mobile device (see the configuration in Table 3.5) due to the "out of memory" error. A small version of the highly accurate model with fewer layers can be executed on mobiles, however, it drains much battery power. For example, processing a single 224x224 photo will consume 11.3 Joules as shown in Figure 3.1 (a).

Most recent work [88] proposed model optimizations such as model weights compressed to reduce the resource consumption of running mobile DNNs at the cost of reduced model accuracy. However, the compressed state-of-the-art ResNet DNN [31] model using their techniques still consumes much battery power (53.3% of the battery capacity for processing a 5-minutes video as shown in Figure 3.1 (b)) on modern mobile devices which often have 7-11 Watt-Hours batteries, while its model accuracy is decreased by 18.3% compared to the highly accurate DNN models (see Figure 3.1 (c)).

Constrained resources on mobile: Unfortunately, the constraints on size

and weight of a mobile device limit its battery capacity and hardware adoption. For lithion battery which is commonly used on mobile devices, it takes more than 10 years to double its energy density (e.g., Watt-Hours/kg) [94, 102]. Therefore, battery capacity will still be a serious concern for mobiles (and wearables as well), making it unattractive to host computationally intensive applications in mobile devices.

On the other hand, recent work proposed leveraging mobile GPUs and hardware accelerators to improve the performance of DNNs running on mobiles [83, 88, 103]. However, these hardware components are still power hungry, and not widely available on commodity mobile devices [93]. In this chapter, we focus on building mobile deep processing systems for commodity mobile devices.

Privacy issues on untrusted cloud: To reduce the energy overhead on mobile devices, it is natural to deploy mobile deep processing in a mobile-cloud setting, as more powerful computation resources such as GPUs are available in the cloud. However, such an approach requires mobile applications to upload a user's personal and sensitive data like photos and videos to a remote untrusted cloud server, which may expose mobile users' personal data to unauthorized people. Data encryption can be used by mobile applications to alleviate the privacy issue, but recent studies have reported that the user data could still be leaked during the process of data processing and analytics on untrusted cloud although the data has been encrypted before it is uploaded to cloud [99, 101, 104, 105]. Therefore, exposing users' raw data on untrusted cloud would still give malicious programs chances to steal personal user data, which is unacceptable for many mobile applications such as mobile payment. In addition, the techniques used for protecting user data introduces additional energy overhead on mobiles, e.g., the mobile-cloud scheme with encryption

enabled consumes 30.3% more battery power compared to the corresponding scheme without encryption as shown in Figure 3.1.

Leveraging DNN features for privacy: A deep neural network usually has multiple layers and each layer will generate a set of features for the next layer. These output features are transformed from the input features and model-specific. These features provably hide the information in the original input, which inherently preserves the privacy of the input data. As deeper layers are performed, stronger privacy will be achieved in their output features as shown in Figure 3.3. In MOCA, we take the features generated by DNN layers running on mobile devices as the input for further DNN processing on the cloud, thus avoiding the need of uploading users' sensitive and personal data. More details will be discussed in § 3.4.

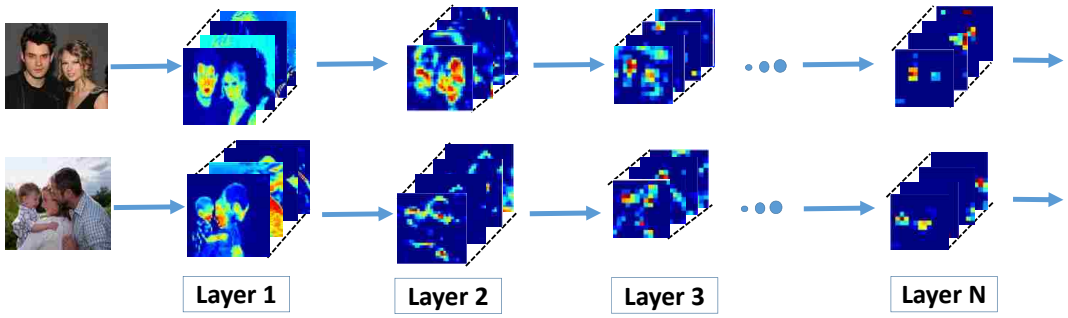


Figure 3.3: Illustration of feature maps extracted in different layers. The visualized feature maps show that the content details on images decrease along with the increasing layers. For example, feature maps generated from the first layer still contain the facial information. However, when the CNN model moves to deeper layer, these facial details disappear.

3.3 System Model and Threat Model

3.3.1 System Model

In our design, we consider three major entities: the cloud server, the app developer and users. The app developer outsources trained deep learning models to the cloud server. Users have mutual authentication capability with the app developer, and can submit a data processing request (e.g., images, video frames either raw or in a transformed/encoded form) to the cloud server. Upon the request, the cloud server can process the received data with a trained deep learning model. Finally, with the ciphertexts of processing result (e.g., classification label) returned by the cloud, users can use their own keys to recover the result.

3.3.2 Threat Model

Figure 3.4: An example: Cloud server may analyze images sent by a user to learn her private information. Capturing such a raw image may expose the medicine the user is taking and thus infer the kind of disease she is suffering.

In our system, we consider the cloud server to be curious-but-honest, that is the cloud server will follow our protocol to perform the service provider role correctly, but it may analyze user data to learn their private information (an example in Figure 3.4). The developers and mobile users are assumed to be honest, and will not collude with the cloud server. These assumptions are consistent with recent related works [106–108]. Based on available information to the cloud server, we consider the following threat model in terms of the privacy protection of data:

The cloud server only has access to the deep learning model (i.e., the portion deployed on cloud) and the received data for processing. Our system aims at preventing the cloud server from learning sensitive information (face, text, etc.) from received data.

3.4 Design and Implementation

Moca is a distributed mobile-cloud based framework designed for deep processing of vision data for mobile applications. In this section, we present our goals in designing Moca, the key techniques proposed for achieving these goals, and the system implementation.

3.4.1 Design Goals of Moca

We design Moca with three primary goals:

- *Achieving privacy for personal user data:* Mobile deep processing facilitates powerful functionalities for mobile applications. However, users' personal and sensitive information should not be exposed or leaked to third parties. Moca

achieves this goal by leveraging the privacy-preserving feature maps generated by DNN models (§ 3.4.2)

- *Achieving differentiated privacy for different application services:* Different mobile application services may have different requirements on data privacy. MOCA achieves this goal by enabling differentiated privacy and allowing developers to deploy their own privacy evaluation models to guide the feature extraction procedure on mobiles. (§ 3.4.3).

- *Achieving energy efficiency for mobile deep processing:* A lightweight and energy-efficient deep processing system is always desirable to extend the battery lifetime for mobiles. MOCA achieves this with several novel optimizations on feature map extraction and DNN model reconstruction to reduce the workload overhead on mobiles (§ 3.4.4 and § 3.4.5).

The system architecture for MOCA is shown in Figure 3.2. Next, we discuss the details of each key technique used in MOCA as follows.

3.4.2 Preserving Privacy with Feature Map

As one of the most popular deep neural networks, Convolutional Neural Network (CNN) [109] has demonstrated its excellent ability in characterizing the high-level abstractions of visual data and outperforms previous approaches by a large margin in many vision-based tasks, such as object recognition and detection [110–113].

CNN is a feed-forward neural network in which the input data passes through the network layers one by one, until it arrives at the outputs. This hierarchical structure was inspired by the biological process to simulate human visual perception which

uses multi-neuron transformations. The output of each layer in a CNN model is usually named as the **feature map**. At each layer, a convolution filter is applied to the feature maps from the previous layer to generate new feature maps with the following equation:

$$u_j^l = f(\sum_{i=1}^{N} u_i^{l-1} * w_{ji} + b_j^l), \tag{3.1}$$

where u_j^l is the jth feature map extracted from layer l, N is the total number of input feature maps, b_j^l denotes the bias, f is an activation function. Different number of feature maps are generated in each layer l.

After applying the convolution filter at each layer, the visual appearance of generated feature maps becomes more "blurred" than the input image or the feature maps generated in prior layers as shown in Figure 3.3. Such a property in CNN provides certain levels of data privacy. As different neural network models have different model architecture (e.g., the number of layers and branches) and parameters, it is difficult for malicious users to extract meaningful information from the feature maps generated in a deeper layer (e.g., layer 3 in Figure 3.3) without having any knowledge of their corresponding models. Even if the models are compromised, the untrusted cloud cannot regenerate the input data (i.e., user's raw data) using the feature maps, as CNN models often use activation functions (i.e., f in (3.1)) which are nonlinear and hence not easy to be reversed.

MOCA leverages the privacy-preserving feature maps generated by CNN models to protect personal user data for mobile vision applications. Instead of uploading encrypted vision data to a remote cloud server, MOCA transfers only the feature

maps for further deep processing on cloud. The cloud server then feeds the received feature maps directly to the remaining layers of the CNN model to generate the final prediction result (see Figure 4.1).

To further protect any information from being disclosed in the predication results, MOCA encodes any plain text in the prediction results, so that the malicious users on untrusted cloud cannot retrieve meaningful information from the output of CNN models. For instance, we can encode the object recognition result of "car" using a numeric value "23", and only the mobile client has the decoding book to recover the plain text from it. We assume there is no collusion between any mobile user and the cloud.

3.4.3 Differentiated Privacy with Quantification

As illustrated in Figure 3.3, CNN has another property: as the CNN model progresses to deeper layers, less human-interpretable information will be present on the generated feature maps. In other words, there is a general trend that *stronger privacy can be obtained with feature maps extracted from deeper CNN layers.* However, extracting feature maps from a DNN layer on mobile devices is not free, as CNN models are computationally intensive while mobile devices are constrained by limited computation and battery resources(as discussed in § 3.2). Mobile vision application incurs additional performance and energy cost, as more layers are executed on mobile devices.

On the other hand, different mobile application (e.g., face authentication vs. photography recognition) may have different privacy requirements on users' data, and users' data from different sources (e.g., photos either from a user's device or

downloaded from the Internet) may require different treatment in practice. We define these different privacy requirement from mobile applications as **differentiated privacy** in this chapter.

MOCA enables differentiated privacy settings for mobile applications by mapping their privacy requirements to the varying privacy levels provided in CNN models. To precisely control the mapping procedure, it is important to quantify the data privacy of feature maps extracted from different layers. However, this is challenging for two reasons. First, mobile application developers can set different privacy requirements for users' data. Thus, it is impossible to define a unified privacy criterion for all mobile applications. Second, varying CNN models have different numbers of layers, and the achievable data privacy levels at different layers are unknown apriori.

To overcome this challenge, MOCA provides a privacy quantification framework running on a cloud server. It allows application developers to customize their own privacy criteria for quantifying the privacy level (e.g., the possibility that the face on the feature maps is recognized as shown in Figure 3.3) for feature maps extracted from a specific CNN layer. The privacy quantification framework works as follows:

First, mobile application developers upload the CNN model used for their applications (App-CNN) and the CNN model used for evaluating their defined privacy criteria (Eval-CNN)hosted in the cloud.

Second, the App-CNN model is run with a large validation image dataset on the cloud server, and the feature maps generated from its each CNN layer are collected.

Third, for each validation image, its extracted feature maps for each App-CNN layer are taken as an input and processed by the first layer of the Eval-CNN model. The predication results are then collected.

Fourth, the privacy quantification framework performs privacy level calculation based on the privacy criteria defined by application developers. For instance, for an object recognition task, a developer can define the privacy level as follows: as long as one of the feature maps extracted from a validation image at a particular layer can be recognized, it is considered a true positive. The privacy level is then defined as the ratio between the number of true positive images over the total number of images in the validation dataset. The calculated privacy level value for each App-CNN layer is stored in an encrypted file (defined as LayerProfile). The LayerProfile which contains the quantified privacy level at each App-CNN layer, will be sent back to application developers and registered in their mobile applications for future reference (discussed in § 3.4.5). We present how we evaluate the privacy levels for various vision tasks in § 3.5.

Possible Dynamic Privacy Adjustment

The privacy quantification framework provides a reference privacy level for each layer of the CNN model. This value is computed as the average of all measured privacy levels based on an appropriate validation dataset provided by application developers. However, as new query data is generated or collected from users, the actual privacy level might be different from the reference privacy level. Therefore, it would be plausible for some application developers to dynamically adjust the model deployment (i.e., the number of CNN layers executed on the mobile devices) to meet the privacy requirement at run-time. Such a dynamic adjustment can be achieved by measuring the run-time privacy level using users' recent query data. If the measured run-time privacy level is worse than the reference level, more CNN layers should be executed on the mobile devices. We leave the design of the dynamic

Method Name	Size Reduction	Speed-up
Tucker Decomposition	√	√
Convolution and Deconvolution	√	
Scalar Quantization and Huffman Coding	√	
Parameter Free Layer Merging		√
Branch Reduction		√

Table 3.1: Optimizations for feature map extraction.

privacy adjustment for our future work.

3.4.4 Feature Map Extraction and Optimization

Moca allows mobile applications to extract feature maps from different layers of a trained deep learning model for the privacy tradeoff. A naive way of producing such feature maps is to run a specified number of CNN layers from an unoptimized CNN model on a mobile device and send the feature maps from the last layer run on the device to a remote cloud server.

However, such a solution incurs large performance and energy overheads in terms of computation and networking resource consumption, which is unacceptable on mobile device: (1) executing a few CNN layers could take several hundred milliseconds on modern mobile phones; (2) the extracted feature maps could be much larger than the input image. For example, the size of the feature maps extracted from the first layer of a ResNet based object recognition model is 3 MB, while the size of an input image is 73.5 KB. To optimize the feature map extraction procedure, we propose five techniques as shown in Table 3.1. Each technique is discussed as follows.

Number of Feature Maps	Size Reduction	Accuracy Loss
4	16x (93.75%)	2.64%
5	12.8x (92.19%)	1.25%
6	10.6x (90.57%)	0.44%
7	9.1x (89.01%)	0.35%
8	8x (87.50%)	0.18%
9	7.1x (85.92%)	0.14%
10	6.4x (84.38%)	0.11%
11	5.8x (82.76%)	0.06%
12	5.3x (81.13%)	0.01%

Table 3.2: Optimized tucker decomposition: it reduces the number of feature maps and their size dramatically with small accuracy loss, for a layer in ResNet-50 object recognition model. The layer originally generates 64 feature maps.

Tucker Decomposition: Traditionally, tucker decomposition was used to decrease the storage size of CNN models by reducing the number of weights in a CNN layer. We find that such a technique can also be leverage to reduce the number of extracted feature maps for a convolution layer.

The number of extracted feature maps for a convolution layer is determined by the shape of the layer's convolution filter which is a 4-dimensional matrix of size $D \times D \times S \times T$, in which $D \times D$ is the size of covered pixels of an input feature map, S is the number of feature maps generated from the previous layer, and T is the number of feature maps generated from this layer. We perform a Tucker-1 decomposition to decompose it into two convolution filters Z (with size $1 \times 1 \times S \times R$) and Y (with size $D \times D \times R \times T$), Since R is much smaller than T, the first convolution filter Z is designed to dramatically reduce the number of feature maps. We demonstrate the efficiency of applying the tucker decomposition technique on a layer in ResNet-50 object recognition model in Table 3.2.

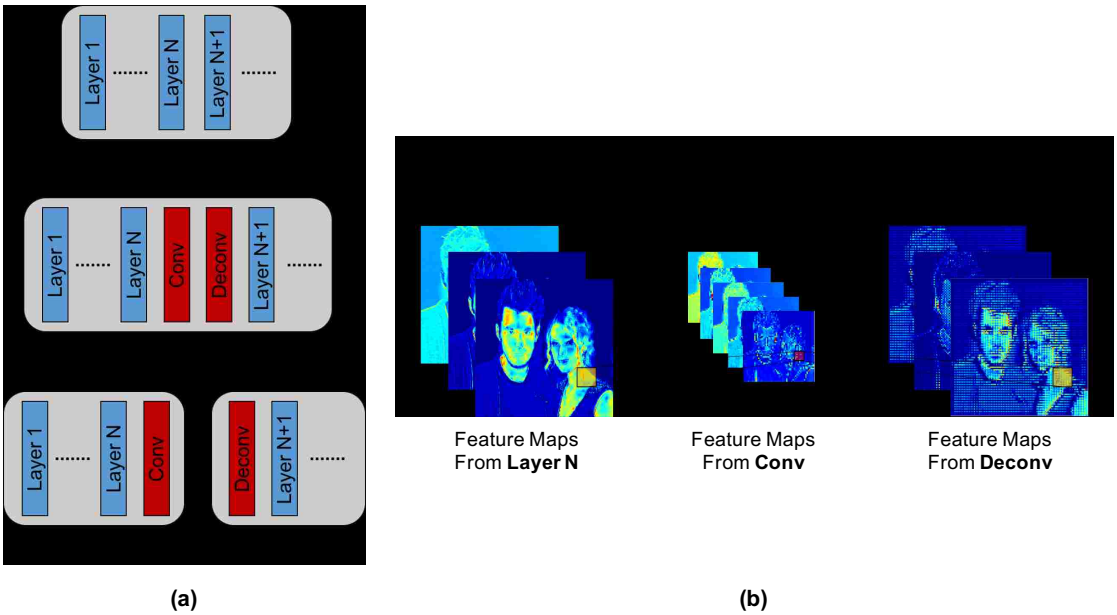


Figure 3.5: A case study of convolution and deconvolution: the convolution encoder reduces the feature size from 112 x 112 x 3 to 56 x 56 x 5 (2.4x reduction).

Convolution and Deconvolution: For some CNN architectures, the resolutions of the feature maps generated by the first few layers could still be high. Hence, merely reducing the number of generated feature maps using Tucker Decomposition does not guarantee small feature map sizes. For example, the resolution of the feature maps in VGGNet's first convolution layer could be as large as 600 x 600 for a detection task, hence, each feature map will be as large as 351 KB. Since multiple feature maps are generated at each layer, large networking overhead is incurred as the maps are sent to the remote cloud.

For CNN models which take images of high resolution, we propose to insert a pair of convolution layer and deconvolution layer in between layers after the 1st Tucker decomposed convolution layer to reduce the size of feature maps at each layer. Figure 3.5 demonstrates an example of how one can reduce the resolution of

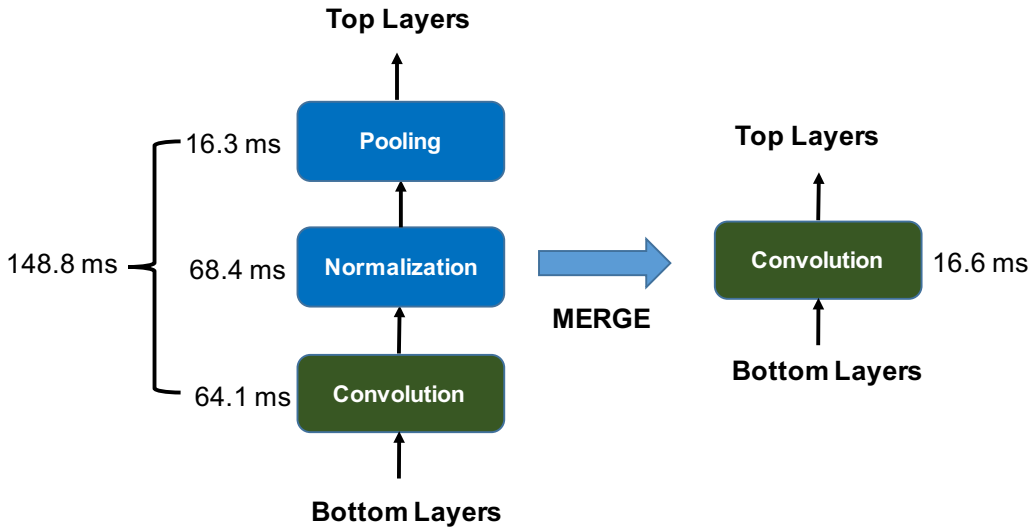


Figure 3.6: Benefits of parameter-free layer merging. The latency is measured using Caffe on Samsung Galaxy S5.

feature maps using the newly inserted pair of convolution and deconvolution layers. The convolution layer reduces the feature maps' resolution (like an encoder) and it is this compressed information that will be sent to the remote cloud if the model configuration determines that feature maps of this layer should be sent remotely. Then, the deconvolution layer running on the remote cloud only recovers the resolutions (not visual appearance) of feature maps before these maps are fed to subsequent layers running on a remote cloud server to complete the deep processing. By fine-tuning [114] the inserted layers, the accuracy loss can be minimized.

Parameter-Free Layer Merging: The state-of-the-art deep learning models incorporate parameter-free layers (i.e., do not have any learned parameters) such as pooling layers or normalization layers. Although they do not have parameters, their executions still consume a significant portion of the total time of running a model. Based on our experiments with ResNet, 40% of the execution time is spent on these

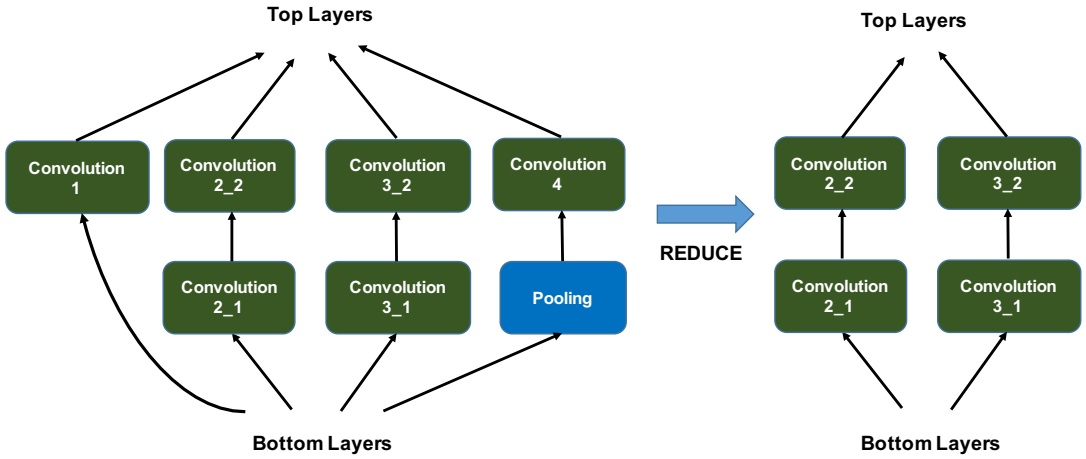


Figure 3.7: Branch Reduction Example on Google's Inception Network. Two branches are reduced.

parameter-free layers.

With merging a parameter-free layer to its preceding (bottom) convolution layer with fine-tuning, a significant speed-up can be obtained with negligible or almost no accuracy loss. This is because the weights in a convolution filter is sparse [115] (i.e., a significant number of zeros), therefore a denser convolution filter can be trained. The new convolution layer functionally performs as both the convolution and the parameter-free layers. As shown in Figure 3.6, 9x speed-up with 0.1% accuracy loss can be achieved by merging a normalization layer and a pooling layer.

Branch Reduction: Modern CNN model may have multiple branches in a network layer such as ResNet or Google's Inception Network (see Figure 3.7). To extract the feature maps from such layers, all the branches should be considered to avoid the accuracy loss. In this case, the size of the extracted feature maps becomes much larger (depending on the number of branches). Similar to parameter-free layer merging, we reduce the number of branches by merging nearby branches based on

the sparsity of a convolution filter. Based on our experiments with a variety of CNN models, the model accuracy is reduced by 0.05% after the branch reduction.

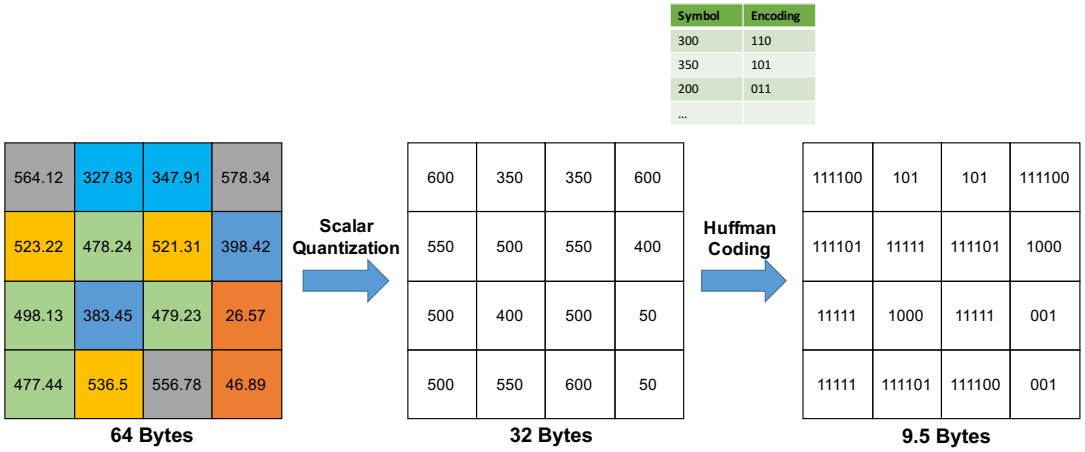


Figure 3.8: An example of Scalar quantization and Huffman coding. 6.7x (85%) feature size reduction is achieved.

Scalar Quantization and Huffman Coding: To further reduce the feature map size, we use a combined scalar quantization and Huffman coding [116] technique to condense the values of the extracted feature maps. As shown in Figure 3.8, the values of feature maps are in floating-point formats.

The idea of scalar quantization is to enable weight sharing so that the feature map values can be represented using a finite set of evenly-spaced values in an encoding book. Since different quantized values in the book have different frequencies, the values of feature maps can be further compressed using Huffman coding. Table 3.3 demonstrates the efficiency of the proposed approach on one of ResNet-50's convolution layer for object recognition tasks. We can achieve 4.87-7.51x (79.47%-86.68%) size reduction. Moreover, the encoding ensures that the data privacy for feature maps is preserved during their transmission over the network.

Book Size	Size Reduction	Accuracy Loss
100	7.51x	1.74%
200	6.09x	0.32%
300	5.49x	0.16%
400	5.12x	0.10%
500	4.87x	0.06%

Table 3.3: Scalar quantization and Huffman coding for a layer in ResNet-50 model for object recognition.

Furthermore, to enable fast element-wise quantization, given a book size, we first rescale the feature map data so that the values in the encoding book are rounded off to the nearest hundred integer value. Then, we perform an element-wise operation over the scaled feature map data with $floor(data/100) * 100$. Different from the four optimization techniques discussed above, scalar quantization and Huffman coding do not change the CNN model architecture, they are applied only after the feature maps have been extracted.

With the techniques discussed in prior sections, we now describe how they work together to reconstruct a more energy-efficient model given any CNN model for mobile vision applications.

3.4.5 Adaptive Model Reconstruction

Developers for various mobile vision applications have different privacy needs. Thus, each developer desires to reconstruct an adaptive model to meet his need. We assume that such model reconstruction is a one-time offline task performed on a GPU server which is different from the untrusted cloud that deploys any adapted train model. Thus, the untrusted cloud cannot access any training data (images, labels, etc).

Given a trained CNN model for mobile vision applications, MOCA integrates

Algorithm 5 Model Reconstruction

Input: D = a pre-trained deep learning model
Output: R = the reconstructed deep learning model
B = the set of encoding books for each processed layer

$B \leftarrow \emptyset$ AND $R \leftarrow D$
$p_{curr} \leftarrow 0$ ▷ The privacy level is initialized to 0.
$l_{proc} \leftarrow l_1 fromD$ ▷ The process starts from the 1st layer.
$A_{comp} \leftarrow 0.1\%$ ▷ Max accuracy loss per layer for compression
$A_{encode} \leftarrow 0.1\%$ ▷ Max accuracy loss per layer for encoding

while $p_{curr} > 0\%$ **do**
    $a_{loss} = 0$ ▷ The accuracy loss for current layer.
    for $f_{num} = 12, -1, 6$ **do** ▷ Tucker decomposition.
        $TuckerDecomposition(l_{proc}, f_{num})$
        $updateLoss(a_{loss})$
        if $a_{loss} > A_{comp}$ **then**
            break
    $decomposeLayer(l_{proc})$
    for $f_{num} = 18, -6, 6$ **do** ▷ Convolution-Deconvolution.
        for $Stride = 2, 1, 4$ **do**
            $ConvDeconv(l_{proc}, f_{num}, Stride)$
            $updateLoss(a_{loss})$
            if $a_{loss} > A_{comp}$ **then**
                break
    $addConvDeconv(l_{proc})$
    if l_{proc} followed by parameter free layers **then**
        $parameterFreeLayerMerge(l_{proc})$
    if l_{proc} has multiple branches **then**
        $branchReduction(l_{proc})$
    $l_{proc} \leftarrow l_1.Next()$
    $p_{curr} \leftarrow evaluatePrivacy()$
    $c_{acc} \leftarrow evaluateComputingCycles()$

for each processed layer l_{proc} **do**
    $a_{loss} = 0$ ▷ The accuracy loss for current layer.
    for $book_{size} = 500, -100, 100$ **do**
        $HuffmanCoding(l_{proc}, book_{size})$
        $updateLoss(a_{loss})$
        if $a_{loss} > A_{encode}$ **then**
            break
    $B.append(getBook(l_{proc}))$

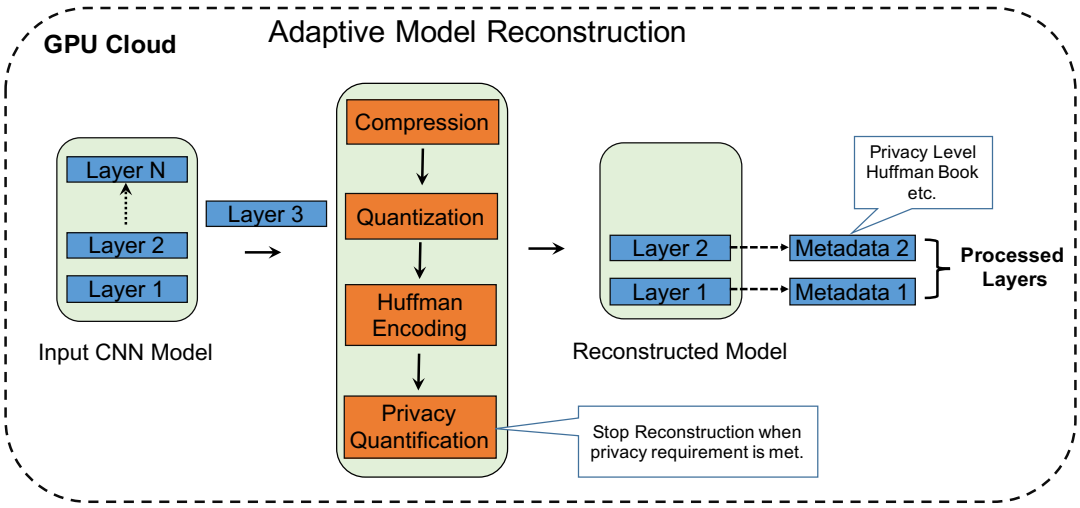


Figure 3.9: Overview of Adaptive Model Reconstruction.

the privacy quantification framework and the feature map extraction techniques (for mobile application's performance) into the model by reconstructing it. Unlike prior model optimization work [88] which requires a fully reconstruction of a trained CNN model since every layer needs to be optimized, MOCA focuses only on the first few CNN layers that are executed on mobile devices, while the architecture for the rest of the majority layers (which will be deployed on cloud with GPUs) remain unchanged.

To achieve more desirable feature size reduction and feature map extraction speed-up, more accuracy losses will result in the CNN models. We propose an adaptive reconstruction algorithm (Algorithm 5) to precisely control the model reconstruction procedure to ensure that the best tradeoff is performed on each layer.

Our model reconstruction algorithm processes a given CNN model layer by layer (Figure 3.9), and for each layer, it tries to minimize the extracted feature size while ensuring that the accuracy loss is kept within a threshold (0.1% by default in MOCA).

`[ReconstructedModel, LayerProfile] ReconstructModel( ModelObj rawModel, TrainDataSet trainImages)` /*CNN model reconstruction*/
`FeatureObj ExtractFeature(Image inputImage, ModelObj cnnModel, LayerID execLayer )` /*Extract the feature maps from execLayer in cnnModel*/
`FeatureID UploadFeature( FeatureObject ftrObj, LayerID execLayer )` /*Upload a feature map to cloud*/
`LayerID SetPrivacy( Float privacyLevel)` /*Set the privacy level*/
`Result DeepProcessFeature( FeatureID ftrID )` /*Deep process a feature on cloud*/

Table 3.4: MOCA API

Once one layer is reconstructed, the privacy quantification framework (see § 3.4.3) will be used to evaluate its privacy level based on the privacy criteria defined by mobile application developers. The adaptive reconstruction procedure stops when a reconstructed layer's quantified privacy meets the privacy requirement set by application developers. Note that some parameters (e.g., line 9 and line 15) in the Algorithm 5 are defined based on our extensive evaluation on a variety of popular CNN models.

After model reconstruction, MOCA places the metadata for each reconstructed layer in an encrypted file (LayerProfile in § 3.4.3) which contains: (1) its reference privacy level; (2) the Huffman encoding book.

3.4.6 Implementation Details

MOCA provides two sets of APIs: one for deep learning model reconstruction and one for mobile application development. These interfaces are compatible with the

Amazon's Android SDK for machine learning [117].

For model reconstruction, we provide APIs for mobile application developers to reconstruct their provided model *rawModel* into a *ReconstructModel*. The *ModelObj* includes both the model architecture definition and the pre-trained parameter weights. MOCA supports deep learning framework Caffe. It allows users to provide their own training dataset for fine-tuning their models in the reconstruction procedure. Several popular data format types such as LMDB, HDF5 and raw image are supported in MOCA. Through the model reconstruction API, the LayerProfile for all processed layers are returned along with the reconstructed model to application developers.

The second set of APIs are defined for mobile application development and deployment. First, the developers use *SetPrivacy* to specify their privacy requirement for specific application or data source. A *LayerID* is returned to indicate which layer of a CNN model will be executed on the mobile device. The *ExtractFeature* function runs on the mobile to extract feature maps from the CNN layer *LayerID*. *UploadFeature* is called to send feature maps to the cloud for further deep processing from the next layer after *LayerID* with *DeepProcessFeature*. The result is returned to mobile client. It could be an object ID for recognition or the bounding boxes for detection.

3.5 Evaluation

The goal of our evaluation is to demonstrate that: (1) the performance and energy efficiency of using MOCA in ResNet-50 based CNN model in several CNN-based

Mobile Client	Samsung Galaxy S5
Processor	2.5 GHz quad-core
Memory	2 GB RAM
Storage	32 GB eMMC flash
Network	WiFi 802.11 a/b/g/n/ac
Battery	Li-Ion 2800 mAh
OS	Android 4.4
Cloud Server	Ubuntu 12.04 Server with NVIDIA GTX TITAN X
GPU	3072 CUDA cores with 12 GB RAM
CPU	Intel Xeon 8 cores
Memory	16 GB RAM

Table 3.5: Mobile device and cloud server used for evaluating MOCA in this chapter.

mobile vision applications, including photograph recognition, scene analysis, face authentication and object detection; (2) MOCA has negligible impact on the service accuracy of different applications and neural network models; (3) Different levels of data privacy can be achieved by adaptively adjusting the feature extraction procedure on mobiles, and MOCA provides stronger privacy than the state-of-the-art approach; (4) MOCA works efficiently for different deep CNN architectures, e.g., VGGNet and GoogleNet.

3.5.1 Experimental Setup

We use Samsung Galaxy S5 as the mobile client to run different mobile vision applications. A cloud server configured with NVIDIA's GTX Titan X GPU is used to host deep processing services for these mobile applications. The configurations of both the mobile client and cloud server are shown in Table 3.5. WiFi is used for

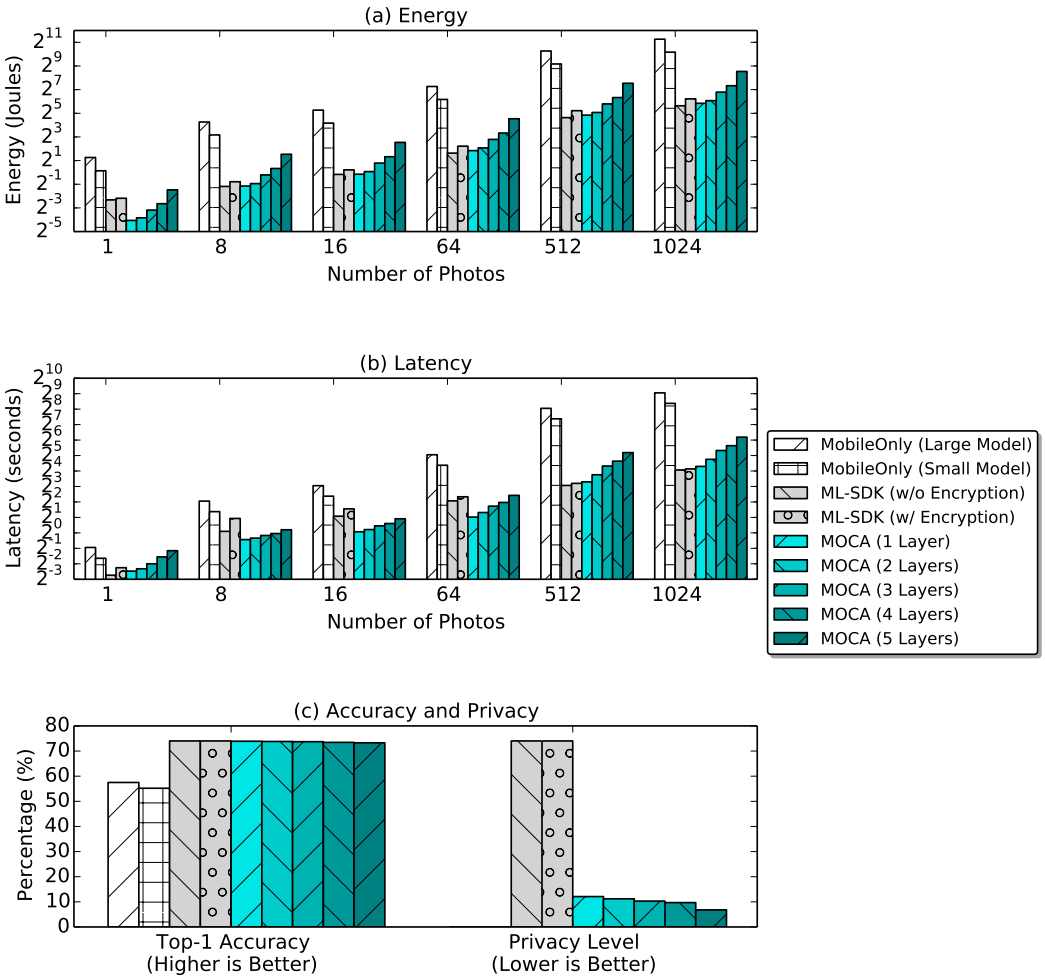


Figure 3.10: Running **CNN-based photography recognition services** with MobileOnly vs. ML-SDK vs. MOCA. The figure demonstrates the (a) energy overhead, (b) latency, (c) accuracy and privacy levels of the photo recognition service. With executing two layers on mobile, MOCA reduces the energy consumption by 17.1-36.7x (94.15%-97.28%) and 1.2-3.4x (16.67%-70.59%) compared to MobileOnly and ML-SDK respectively, and decreases the latency by 6.9-11.0x (85.51%-90.91%) and 1.3-1.9x (23.08%-47.37%) compared to MobileOnly and ML-SDK respectively, while providing much stronger privacy than ML-SDK with the similar service accuracy.

the connection between mobile devices and the cloud server.

To measure the energy usage of the mobile devices when running mobile vision workloads, we instrument the mobile devices' battery-leads to make them draw power from the Monsoon power monitor [118] instead of their batteries. The reported absolute power consumption of running a workload includes the basic power consumption of the mobile device. For the energy usage comparison between different solutions, this base power is subtracted from the power used when running the workload. We compare MOCA with the following state-of-the-art mobile deep processing solutions:

- **MobileOnly:** All the deep processing for mobile vision data is executed locally on mobile devices. As discussed in § 3.2, the deep learning models with higher accuracy either cannot run on commodity mobile phones or consume much more power than other solutions. Therefore, two compressed neural network models that have been optimized specifically for commodity mobile platforms [119], and often used by the deep learning community are used in our evaluation, they are referred to as *MobileOnly (Large Model)* and *MobileOnly (Small Model)*.

- **ML-SDK:** The Android SDK for machine learning released by Amazon [117] is one state-of-the-art approach that allows mobile applications to access machine learning services hosted on a remote cloud server. It requires mobile users to upload their personal data to the cloud. However, it does not provide interfaces for users to protect their data privacy. We extend its interfaces to support hardware encryption on modern mobile devices.

Workload	Privacy Criteria	MOnlyPrivacy	SDKPrivacy	MocaPrivacy
Photography Recognition	The ratio of objects recognized as ground truth.	0.0%	74.0%	11.2%
Scene Analysis	The ratio of scenes recognized as ground truth.	0.0%	53.7%	8.1%
Face Authentication	The ratio of faces not recognized as "other".	0.0%	97.8%	0.0%
Object Detection	The ratio of detected ground-truth objects.	N/A	63.5%	1.2%

Workload	MOnlyACC	SDKACC	MocaACC	Energy Saving
Photography Recognition	57.8%	74.0%	73.8%	3.4x (70.59%) & 36.7x (97.28%)
Scene Analysis	45.3%	53.7%	53.6%	1.2x (16.67%) & 27.1x (96.31%)
Face Authentication	86.7%	94.6%	94.3%	2.1x (52.38%) & 24.8x (95.97%)
Object Detection	N/A	63.5%	63.1%	6.7x (85.07%) & N/A

Table 3.6: Summarized results. The privacy values, the lower the better. For accuracy values, the higher the better. The energy efficiency column shows the energy saving of Moca compared to ML-SDK (left) and MobileOnly (right). **MOnly:** MobileOnly; **SDK:** ML-SDK; **ACC:** Accuracy.

To evaluate the service accuracy and data privacy provided by different solutions, a set of common deep learning methods [1] are performed against the uploaded data from the mobile application to verify whether any information could be leaked or exposed from these data. As different applications may have different criteria for data privacy, a variety of neural network models are used to evaluate their privacy levels. Table 3.6 illustrates the common privacy criteria used in these mobile vision applications and how Moca achieves privacy protection with significantly reduced energy cost and minimum accuracy loss.

3.5.2 Photography Recognition

We first evaluate Moca with photography recognition applications on a Samsung S5 mobile device. The photograph recognition applications leverage CNNs to recognize objects that appeared on photos. We use ImageNet [120] 1000-class object recognition dataset for evaluations. This dataset has 1.28M training images and 50K validation images. Each image is preprocessed into an image with a resolution of 224 x 224 pixels, and each pixel requires 1.5 bytes for YUV representation. The

[1] Unless otherwise noted, we use the state-of-the-art ResNet [31] (more specifically the most widely used 50-layer ResNet-50) as our CNN model architecture for evaluation.

preprocessed photos are used as the input for CNN models. When multiple photos are issued for deep processing, the transferred data are batched to achieve network efficiency in ML-SDK and Moca.

For MobileOnly, two compressed object recognition models [119] from ResNet-50 are used. Its corresponding uncompressed model with higher accuracy is deployed in a cloud server for ML-SDK. Moca reconstructs the corresponding uncompressed version of the object recognition model and deploys its first five layers on mobile and the remaining layers on cloud server. Note that the model reconstruction in Moca is conducted on a cloud server in offline mode, hence, it does not affect the performance of the application services. More CNN layers can be deployed on mobiles based on users' privacy requirements and the battery budgets available in mobiles.

In terms of energy usage on mobiles, MobileOnly consumes the most battery power compared to other two solutions as shown in Figure 3.10 (a), although its models have been optimized for mobile platforms. This is because the neural network models are computationally intensive, which requires large memory and computation resources. When Moca executes one or two layers on mobile and transfers the extracted feature maps to a remote cloud server for further deep processing, it consumes up to 36.7x and 3.4x less power than MobileOnly and ML-SDK respectively, since its computation workload is light, and the size of the extracted feature maps is 8x smaller than the size of preprocessed photos (which reduces the energy overhead on networking). As more layers are executed on mobile, the power consumption of executing CNN layers will offset the reduced energy on networking.

Moca also reduces the latency of the object recognition services by up to 11.0x

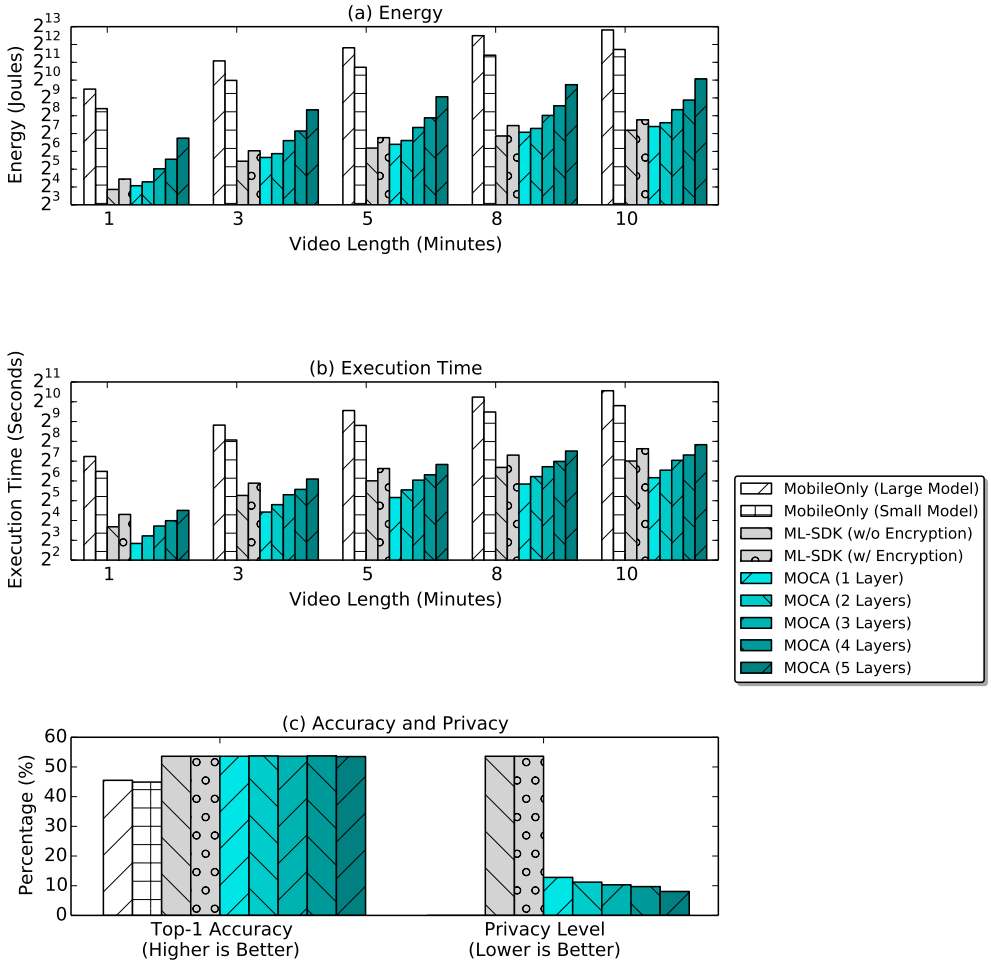


Figure 3.11: Running **CNN-based scene analysis** on videos with MobileOnly vs. ML-SDK vs. MOCA. With executing two layers on mobile, MOCA reduces the energy consumption by 27.1x (96.31%) and 1.2x (16.67%) compared to MobileOnly and ML-SDK respectively, and decreases the latency by 12.9x (92.25%) and 1.7x (41.17%) compared to MobileOnly and ML-SDK respectively, while providing much stronger privacy than ML-SDK with the same service accuracy.

(90.91%) and 1.9x (47.37%) compared to MobileOnly and ML-SDK respectively when two layers are executed on mobile as shown in Figure 3.10 (b), since less data is transferred via the networking. As cloud server has much more powerful resources, the execution time of deep processing a photo on it takes 16.2 ms on average, which is a small portion of the service latency. As more layers are executed on mobile for stronger privacy, their execution times on a mobile client will become a significant portion of the end-to-end service latency.

To evaluate the privacy levels of the compared solutions, a competitive and highly accurate CNN model for object recognition is executed using the data uploaded by ML-SDK and Moca respectively on the cloud server. We quantify the privacy level by using the recognition ratio reported by the CNN-based object recognition model. As Moca extracts feature maps from deeper layers, stronger privacy level will be achieved. For feature maps extracted from the second layer, Moca's privacy level is 11.2%, which is much lower than 74.0% (ML-SDK's privacy level) as shown in Figure 3.10 (c). When Moca extracts feature maps from the fifth layer, its privacy level reaches 6.8%, which means that the possibility that the objects on the preprocessed photos can be identified by highly accurate object recognition models is small.

With cloud servers, ML-SDK and Moca can use highly accurate neural network models to provide higher service accuracy than MobileOnly. Most importantly, using Moca only has negligible impact on the service accuracy.

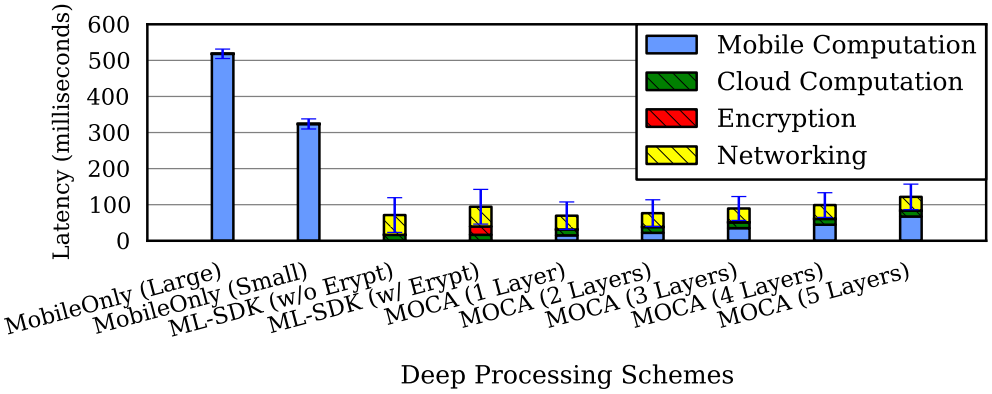
3.5.3 Scene Analysis

Intelligent scene analysis enables mobile devices to support new vision-based applications such as augmented reality and video games. It helps mobile applications recognize the scene displayed on photos and videos. The scene analysis model is trained using the 2.5 M Places [121] dataset which includes 205 different scene categories. For validation, we collect a dataset of 50 video segments that covers 117 of the 205 recognizable scenes and each video is about 10-minute long. These videos are recorded with the frame rate of 30 FPS (frames per second), and 5 frames in each second are preprocessed into images of 224 x 224 dimensions for further deep processing. Two compressed neural network models for scene recognition from ResNet-50 are used for MobileOnly, and its corresponding highly accurate uncompressed model is used for ML-SDK and Moca.

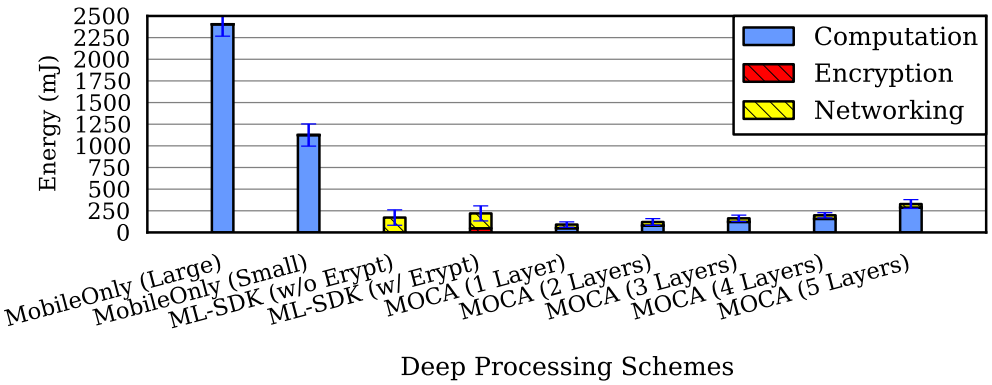
Similar to the trends on energy usage and service latency discussed in § 3.5.2, Moca performs 27.1x (96.31%) and 1.2x (16.67%) more energy efficient than MobileOnly and ML-SDK respectively, when feature maps are extracted from the second neural network layer. The results of quantifying the privacy level and service accuracy with a highly accurate scene recognition model demonstrate that Moca provides stronger privacy (recognition ratio: 8.1%) than the state-of-the-art solution ML-SDK (recognition ratio: 53.7%), while guaranteeing the same service accuracy.

3.5.4 Face Authentication

Face authentication is becoming a popular function that allows mobile applications to verify a user's identity via facial recognition. In this experiment, we follow the pipeline of deep face recognition [122] to train our CNN facial recognition model.



(a) Latency Breakdown



(b) Energy Breakdown

Figure 3.12: The latency and energy usage breakdown of deep processing a photograph for face authentication.

The dataset we use is the FaceScrub [123] dataset which includes over 100K face images of 530 People, and a distraction set of 1000 images for "other" people. The output of this model is either one of the 530 people or "other" if the model does not recognize the input face.

To understand the energy usage and end-to-end latency for such an application

service, we break down its procedure into several components as shown in Figure 3.12: computation on mobile, computation on a cloud server, data encryption on mobile and networking time for user data transfer. It takes 323.8 ms and 518.2 ms for MobileOnly to perform the face authentication with the large and small neural network models respectively. For ML-SDK which relies on a cloud server for deep processing (15.2 ms for processing one facial photo), a large percentage of the latency is contributed by data transfer over the network. MOCA adds overhead on mobiles due to the feature extraction procedure, however, it reduces the networking overhead because its extracted feature map size is much smaller than the size of input images for CNN models. As shown in Figure 3.12(a), MOCA provides lower latency than ML-SDK (with encryption enabled) when fewer than four neural network layers are executed on mobiles.

As for energy usage for processing facial authentication of one photo, MobileOnly consumes 1.1 - 2.4 Joules, while ML-SDK consumes 0.2 Joules in which 82.3% is caused by networking. For MOCA, most of its energy overhead comes from feature extractions. Extracting the feature maps from the second layer consumes 0.1 Joules. Based on the privacy criteria defined in Table 3.6, MOCA will not leak any meaningful information in the facial photo, even if the untrusted cloud uses a highly accurate CNN approach to recognize the feature maps in MOCA.

3.5.5 Object Detection

Different from object recognition, the object detection is more computationally intensive, as it has to locate an object in an image using a bounding box and also recognize what the object is. The typical examples include pedestrian detection,

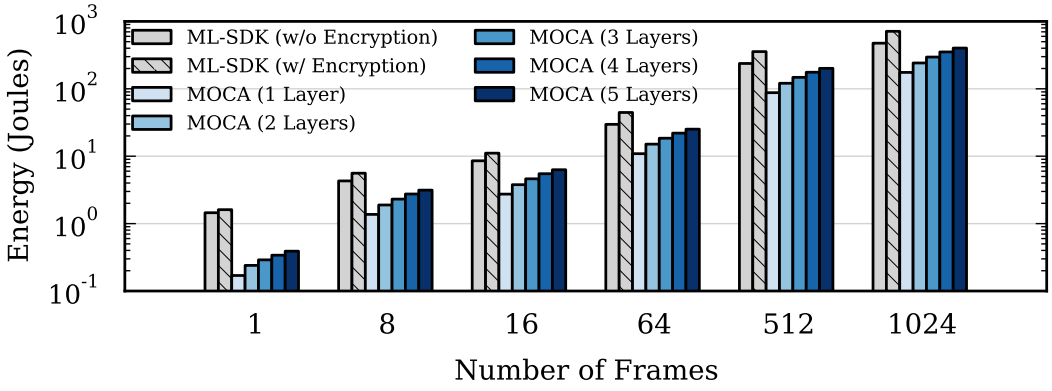


Figure 3.13: Running **CNN-based object detection** on video frames with ML-SDK vs. MOCA. MOCA reduces the energy usage on mobile by 1.9x - 6.7x (47.37%-85.07%) compared to ML-SDK.

road sign detection and so on. The accuracy of object detection is measured using mean average precision (mAP) [124].

The object detection method we use is Faster-RCNN [33] that is built on top of ResNet-50. We use the ImageNet video object detection dataset for evaluation. The input image resolution of our model is 600 x 800. The object detection CNN model takes each frame of the video as its input and detects 30 categories of objects in the video frames.

As shown in Figure 3.13, MOCA is 1.9x - 6.7x (47.37%-85.07%) more energy efficient than ML-SDK, because MOCA reduces the data transfer size dramatically by 8x (87.5%) compared to ML-SDK, and consumes less energy on networking operations.

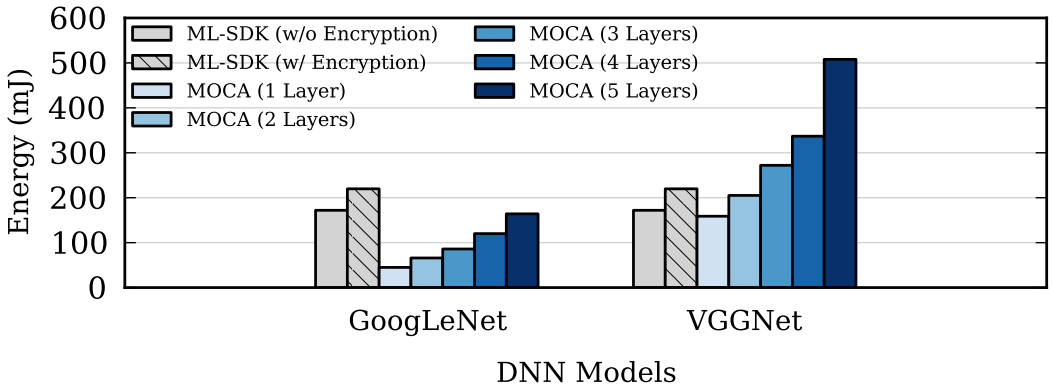


Figure 3.14: Energy usage of running different DNN models with ML-SDK vs. MOCA.

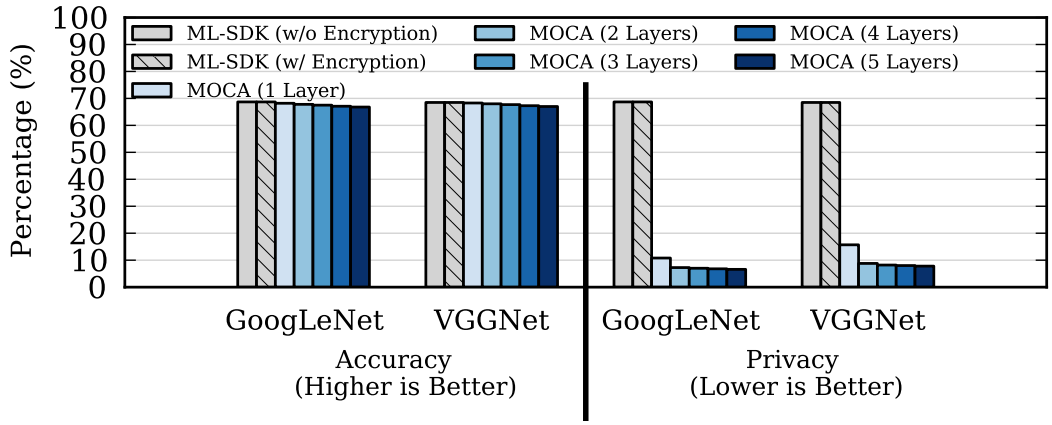


Figure 3.15: Service accuracy and privacy evaluation of using ML-SDK vs. MOCA with different DNN models.

3.5.6 Applicability in Different CNN Architectures

In this section, we evaluate MOCA with different CNN model architectures to demonstrate its effectiveness. We use two highly accurate object recognition models from GoogLeNet [125] and VGGNet [126] respectively. The photos in the ImageNet dataset are used as the input for these models. As we expected, MOCA has lower energy overhead than ML-SDK when a few neural network layers are executed on

mobiles. As deeper layers are executed, more energy is required. Such trends are seen in Figure 3.14 as we apply MOCA in both GoogLeNet and VGGNet models.

Besides the energy efficiency, MOCA has negligible impact on the service accuracy for different neural network models, and provides stronger privacy for both GoogLeNet and VGGNet models, as shown in Figure 3.15. Moreover, our results from the experiments with different neural network models for a variety of vision applications discussed in § 3.5 provide strong evidences for the effectiveness of MOCA.

3.5.7 Privacy Discussion

The privacy protection discussed in the chapter is a pioneer work toward a fully secure system. There are possible ways to compromise the privacy claimed under stronger threat models.

Training Model for Attacks: An active malicious cloud server may train an attacking model using feature maps instead of raw images. For example in a face recognition system, though a face recognition CNN model trained using raw images cannot correctly recognize faces in received feature maps, it can recognize accurately if the CNN model is trained using labeled feature maps. However, there are some difficulties for malicious clouds to launch such attack. First, without knowing the feature map extraction module (i.e., the portion of CNN deployed on mobile), the attacker cannot generate the required feature maps for training. Second, even if the cloud colludes with mobile users so that they can generate the required feature maps, it is still hard to obtain enough training data without having access to the training datasets.

Deriving Raw Images: It would be more vulnerable if the cloud can derive

raw images from received feature maps. A possible way to launch such attack is to deploy the recently emerging generative adversarial nets (GAN) [127]. This also requires mobile users to collude with the cloud server so that they have the access to the feature map extraction module. However, the research for GAN is still in an early stage and the quality of derived raw images may still be questionable. We would like to investigate such attack in our future work.

3.6 Related Work

Deep processing for mobile vision: Convolutional neural network (CNN/ConvNet) is a type of feed-forward artificial neural network, which has been widely used for processing images and videos [91, 125, 126]. Since a CNN model achieves higher accuracy, it has been ported to run on commodity mobile platforms such as Android [87, 128, 129] and iOS [130]. For example, the recent popular mobile applications like Pokemon Go and Google's Tango [131] have already used neural network technologies for processing mobile vision.

However, processing vision data using any neural network technique on resource-constrained mobile devices is expensive. Several model optimizations such as model compression were proposed [88, 132–134] recently to trade off CNN accuracy for resource usage. Our analysis on those compression methods demonstrate that such techniques still have large performance and energy overheads on mobiles. Meanwhile, majority of those methods work effectively on classical CNN models which have major computations on the last few fully connected layers. However, the state-of-the art CNN models like ResNet [31] and Google's Inception [135] have removed

most of such layers, making those compression methods less effective.

Recent work [83, 103, 136, 137] proposed using GPUs and hardware accelerators like application-specific integrated circuits (ASICs) to accelerate the deep processing on mobiles. However, these hardware devices are not widely available on commodity mobile devices [93]. In this chapter, we focus on building energy-efficient deep processing system for commodity mobile devices.

Mobile Offloading to Cloud: To reduce the burdens on mobile devices, the concept of computation offloading [138] has been proposed for decades to shift the computationally intensive workload to remote cloud servers. Prior work [139–141] mostly focused on offloading for conventional mobile applications. MOCA is designed for new DNN-based mobile applications. It decouples the hierarchical layers of DNN models and splits them between mobile and cloud platforms. The workload on mobile could be dynamically adjusted by MOCA based on the privacy requirements of mobile applications.

Preserving privacy for personal user data: Shifting mobile workloads to remote cloud could empower mobiles' capabilities, however, this runs the risk of leaking users' personal and sensitive data, as cloud servers are shared resources which are untrusted. Many image protection techniques were proposed to support data privacy for mobile vision, such as data encryption [142, 143], and huffman coding [144]. However, the performance and energy usage of mobile vision tasks could be significantly hit by these privacy protection mechanisms. Moreover, the encrypted user data would still be leaked during the data processing on cloud servers [101, 104, 105]. MOCA proposes a novel way to protect data privacy by uploading only the privacy-preserving feature maps of user data into untrusted cloud,

without affecting application services' performance and accuracy. Unlike differential privacy [145] that provides both strong integrity and privacy guarantees, Moca achieves differentiated personalized privacy to meet the requirements of different applications.

3.7 Summary

We present Moca, an energy-efficient and privacy preserving system for DNN-based mobile vision applications. It leverages the feature maps extracted at different layers of hierarchical neural network models to provide highly accurate application services with low energy overhead for mobile devices, and also enables differentiated privacy protection for personal user data. We validate these benefits using a variety of typical mobile vision applications, and demonstrate that Moca is 1.2x - 6.7x (16.67%-85.07%) more energy efficient while providing stronger privacy compared to existing solutions.

Chapter 4

DeepCham: Collaborative Edge-Mediated Adaptive Deep Learning for Mobile Object Recognition

In this chapter, we present DeepCham - the first adaptive mobile object recognition framework that allows deep learning model to be used successfully in mobile environments. Deep learning techniques achieve state-of-the-art performance on many computer vision related tasks but still face visual domain bias problem. To solve this problem, DeepCham is mediated by an edge master server which coordinates with participating mobile users to collaboratively train a domain-aware adaptation model which can yield much better object recognition accuracy when used together

with a domain-constrained deep model. Using a newly created image dataset collected using smartphones from different locations, time of a day, and device types, we show that DeepCham improves the object recognition accuracy by 150% when compared to that achieved merely using a generic deep model. We also demonstrate the feasibility of DeepCham using an implemented prototype.

4.1 Introduction

Contemporary mobile devices (e.g., smartphones, mobile robots) equipped with capable cameras and powerful processors enable many exciting mobile computer vision applications such as augmented reality [41, 146–148], robotic pets [149] etc. The core technique underlying majority of these applications is object recognition, i.e., running computer programs to identify objects in an image or video sequence. Traditional object recognition algorithms operate robustly for specific vision tasks in controlled environments (e.g., wine label reading, OCR-based language translation) but exhibit high variation in accuracy for more general mobile object recognition task in unconstrained mobile environments. The key challenges are: (1) *model inclusiveness:* it is hard to know in advance what objects are available in an environment to pre-train an inclusive recognition model for such an environment, and (2) *visual domain shift:* every mobile visual domain is highly variable: changes in image resolution, lighting, background, viewpoint, and post-processing (or any combinations of the above) can adversely impact the accuracy of computer vision algorithms.

Deep learning method, which learns a multiple-level neural network with very

large datasets, opens a door to solve the challenging mobile object recognition problem. Specifically, the recently advanced Convolutional Neural Networks (CNNs) [29], achieves the state-of-the-art performance on large-scale object recognition tasks, e.g., the 1000-class object classification task in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Such large-scale recognition model (we name it the generic deep model) can be trained to recognize most objects in real-life usage which seems to solve the *inclusiveness* issue. However, a deep neural network trained using such large-scale datasets does not guarantee sufficient recognition rate in each target visual domain (i.e., the context where the deep model is applied to) [38, 39, 150], e.g., it may recognize well the animal specimens displayed in a museum but not those alive in a zoo.

Recently it has been shown that the object recognition accuracy of a generic deep model in a target visual domain can be significantly boosted using supervised deep model adaptation methods [38, 39]. An effective adaptation requires high-quality training instances (good image feature and correct image label) captured from such target visual domain. However, in mobile object recognition task, obtaining such high quality training instances is challenging due to the highly variable mobile visual domains.

In this chapter, we propose an edge-mediated collaborative deep learning adaptation framework, DeepCham, that aims to solve the challenging mobile object recognition problem. DeepCham creates domain-specific adaptation training instances using in-situ photos from participating mobile devices, and enables deep model adaptation in both personal spaces (e.g., home) and public spaces (e.g., museums).

The deep model adaptation in mobile environments should incur minimal human effort, respect user privacy, be resource efficient, and be done in real-time. DeepCham achieves these goals via three system components. The first component is a domain-aware image selection pipeline which is run collaboratively among an edge server [151–154] and all participating mobile devices. It includes a function which identifies on-device mobile photos matching the target visual domain specification, and a distributed image pruning algorithm that allows participating mobile devices to collaboratively select a representative subset of images, considering a per-device constraint on the number of selected images. The second component is a training instance generation pipeline that runs on each participating mobile device. It consists of an object proposal algorithm targeted for fast and accurate object extraction from a mobile photo, and a generic deep learning model to facilitate training instance generation. The deep model takes the proposed objects and automatically generates labeling recommendations to the user. The image features are simultaneously extracted from a hidden layer of the deep model. The third component is an adaptation training pipeline which trains a shallow model (e.g. SVM) using the resulting training instances from all collaborative users.

DeepCham performs object recognition on the target visual domain using a *late fusion* technique similar to the method used in [38, 155, 156], which combines the recognition results of both the generic deep learning model and the adapted domain-specific shallow model. Different from existing works, the *late fusion* method used in DeepCham does not require training a separate domain-specific deep model. Instead, it provides the domain information (i.e., the set of existing classes/labels in the adaptation model) to the generic deep model so that it ignores domain-unrelated

classes in its prediction. (More in **section 4.4.2**)

We evaluate DeepCham using a new collected dataset that enables systematic evaluation of adaptation performance under different mobile visual environments. Using this dataset, we first show that the generic deep learning model has poor performance in unconstrained mobile environments, then we evaluate how environmental factors impact the performance of adaptation models. Next, we evaluate the effectiveness of the three components of DeepCham and their contribution to the end-to-end adaptation accuracy. Finally, we demonstrate the feasibility of DeepCham framework with a prototype system.

In summary, our contributions are as follows:

- An edge-mediated framework for performing adaptation using deep learning models for object recognition under different mobile visual domains.
- An attribute-based mobile visual domain specification method for flexible adaptation task initialization and candidate training image selection.
- Fast, collaborative training instance generation pipeline for adaptation designed to run on mobile devices. Our approach aims at minimizing human effort, respecting user privacy and being resource efficient.
- Creation of a new image dataset which enables controlled experimental evaluation of deep model adaptation of different mobile visual domains.
- A prototype system implementation demonstrating the feasibility of the framework.

4.2 Motivation, Challenges and Related Work

4.2.1 Motivation

The design of DeepCham is based on the following key observation: In the mobile context, there are an abundance of unlabeled (i.e., no text annotation) high-resolution domain-aware photos captured by smartphone users. Nowadays, mobile photos are attached with information-rich metadata, e.g., the EXIF data contains detailed information about the camera, the settings of the shot, the date/time, location etc. Such metadata can be easily extracted using standard libraries from modern programming languages. In addition, it is also easy to extend the scope of image metadata using APIs provided by other mobile services, e.g., we can get the weather data by sending date/time though APIs of the weather service. On the other hand, the target visual domain for which we want to make adaptation can also be specified using such metadata attributes. Photos that can be potentially used for adaptation training are identified by matching the image metadata to pre-defined target visual domains.

4.2.2 Challenges

Despite the potential of in-situ mobile user photos, using them directly for adaptation training purposes entails several challenges: First, mobile user photos are not labeled and in general contain multiple objects. This requires solving a hard object detection and recognition problem, i.e., identifying both the location and the class/label of multiple objects in images taken by mobile phones. Second, multiple users are desirable to cover a more inclusive set of objects in the target visual

domain. This entails coordination challenges of users using a different label for an object class and addressing the image redundancy problem of users taking similar photos of the target domain. Third, typically deep learning adaptation training is performed in the cloud with high processing power and huge amounts of data. However, in our case, sending the raw mobile user photos to cloud for processing is not attractive due to privacy issues and high uplink bandwidth/energy consumption. Thus an edge-based solution is required which ideally would perform adaptation training on the mobile phones.

4.2.3 Related Work

Visual domain adaptation. For object recognition task, a visual domain is defined as the context of image/video recognition including a variety of visual cues such as visual illumination, background etc. To increase the recognition performance in a particular target visual domain, visual domain adaptation methods have been extensively studied by the computer vision community [157]. Recent work shows that even with the state-of-the-art deep learning models, visual domain shift is still a problem [38, 39, 150], and various server-based approaches have been proposed. In general, existing adaptation techniques are not specific to mobile visual domains, and are not concerned with mobile resource limitations. In contrast, we investigate the case of mobile visual domain adaptation where potentially useful image data exist in mobile phones but should incur minimal human effort while preserving privacy and minimizing mobile resource usage.

Deep learning on mobile devices. Deep learning models have been ported to run on mainstream mobile platforms such as Android [128, 129, 158] and iOS

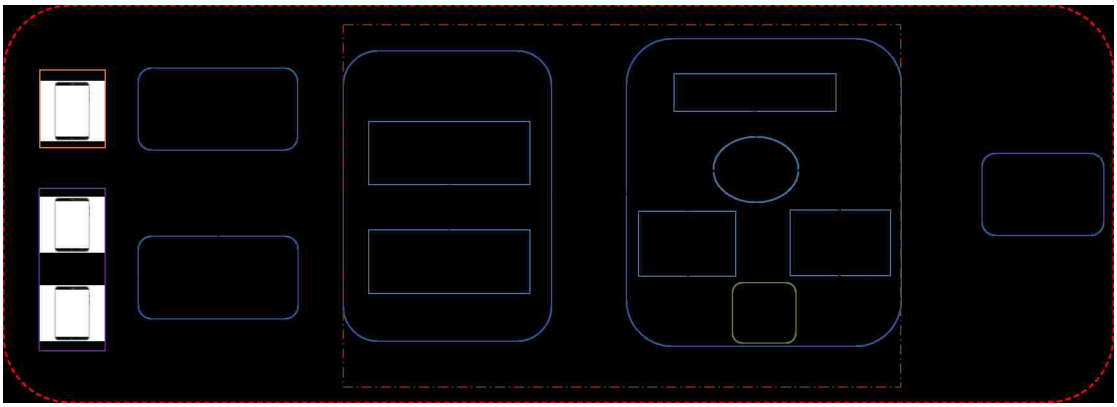


Figure 4.1: The Adaptation Training Overview

[129, 130]. In a recent sensor computing paper, Lane and Georghiev [159] build a system for audio sensing (e.g., speaker identification) using deep learning, showing efficient operation and good accuracy on mobile devices with offline training. In contrast to audio, we show that mobile visual domains require online adaptation training. Furthermore, in our system design, we run the deep learning model on mobile devices for multiple purposes: it is not only used for object recognition, but also for producing high-quality training instances, which requires proper object labels and discriminative image features, in the supervised adaptation training.

Training instance labeling for supervised machine learning. In supervised machine training, the quality of labels assigned to training instances is very important, especially when the number of training instances is very limited. When multiple users are involved in labeling, there is also a label-consensus problem [160] for labeling the same object class. To facilitate the labeling process, interactive labeling techniques [161–164], which combine the power of automatic image labeling and human feedback, have been investigated by the machine learning community.

However, those techniques rely on well-prepared training images instead of raw images from personal mobile devices.

Object bounding box proposal. To automatically identify and crop objects from complex multi-object images, DeepCham relies on the recent advancement of real-time object bounding box proposal algorithms [165, 166] from the computer vision research community. Traditionally, the bounding box proposal algorithms are used as an essential step in known object detection task [32, 166, 167] which requires accurate cropping of target objects (i.e., objects that appear in training images). This means that the algorithm locates many possible bounding boxes around potential objects (usually more than 1000 sorted by confidence score) which are processed later by a computer program to select the best bounding boxes for the target objects. DeepCham involves human users for object labeling, so it cannot afford to use many bounding boxes. In this chapter, we analyze the new requirements and propose an *enhanced bounding boxes strategy* (EBBS) to address this new challenge.

Object recognition in mobile systems. Existing mobile augmented reality (AR) systems [41, 146–148] use sophisticated sensor-based techniques for context inference combined with traditional computer vision techniques for object recognition: they send test images to a GPU-equipped server which extracts image features and matches them to those of annotated object images in a database. These image-matching based systems require exact query images and manual labeling of initial population of annotated images in the database. Our system focuses on adaptation of the state-of-the-art deep learning technique for more generalized object recognition (i.e., it does not require an image database). Furthermore, we show that it can be executed on mobile devices as opposed to GPU-enabled servers.

Edge computing. DeepCham follows the edge computing paradigm [151–154, 168] by pushing computations to the edge of the Internet. The "one-hop" edge server enables the coordination among collaborative users with low-latency communication and sufficient data cache capacity. In addition, it is quite likely that mobile users connected to the same locally deployed edge server (e.g., at a museum) have photos that match the same target visual domain. However one main difference with existing approaches is that DeepCham offloads more privacy-sensitive image processing operations to mobile devices. This fact posed several unique algorithmic and system challenges which we had to address in this chapter.

4.3 DeepCham Adaptation Traning

4.3.1 Training System Overview

DeepCham works by issuing adaptation training tasks to participating mobile devices (Figure 4.1). Each adaptation task targets a particular mobile visual domain, and consists of one initiator, one master and multiple workers. The initiator can be anyone who initializes an adaptation task by specifying the characteristic of the target visual domain and sending such specifications to the master. The master is a software module that executes in a local edge server. Its goal is to accept an adaptation task request and coordinate collaboration among workers for this task. Workers are software modules that reside on mobile user phones. The workers receive tasks from the master and execute the adaptation task by generating and sharing training instances of a target domain based on their captured in-situ images. These instances can be later used to adapt existing generic deep models to the target mobile visual

domain. Subsequently, each user can use this adapted model for object recognition in the target environment.

The Adaptation Training is designed to allow mobile users with common interest to train a domain-aware adaptation model collaboratively. This training procedure can be subdivided into four steps, namely (i) task initialization, (ii) domain-aware image selection, (iii) training instance generation, and (iv) shallow model training.

4.3.2 Task Initialization

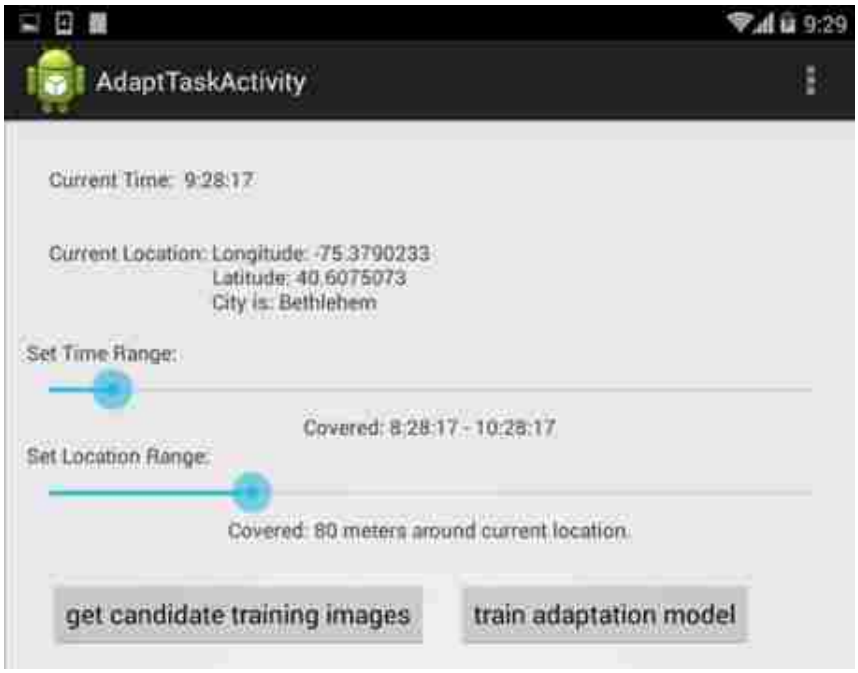


Figure 4.2: An Example "On-site" Adaptation Task Specifications.

The adaptation task is initiated by a user, i.e., the initiator, interested in recognizing objects in a particular mobile visual domain (e.g., creating an adapted model for a personal space with the help of friends) or by a user who wishes to launch a

training campaign for a public space (e.g., creating an adapted model for a museum with help of visitors). The initiator first creates a characterization of the adaptation task and then it sends the specifications to a local edge server.

An adaptation task characterizes the target mobile visual domain by a set of context attributes such as time, location or weather. These attributes aid workers in selecting matching photos they have in their devices. DeepCham allows specifying both remote and on-site tasks. Remote tasks can be initiated by any authorized initiators with no location or time constraint, while on-site tasks are initialized by an onsite user who would like to train and use an adaptation model for her current mobile context. In Figure 4.2, we show a screenshot of an on-site adaptation task specification on an Android device. In this scenario, the client software automatically set the visual domain attributes by calling system APIs (e.g., time and location). It should be noted that the visual domain attributes can be defined at different levels of granularity. For example, the location can be defined at the granularity of places [121] (restaurants, classrooms etc.).

On receiving the task specifications, the master first checks if there is an existing adaptation model matching the specified target visual domain. The model matching criteria can be flexibly determined, e.g., if the model is defined with location attribute as a region ($GPS_{coord} + Radius$), the master may calculate and present the region overlapping ratio with existing adaptation models (if defined with location regions), and based on this information the task initiator can make a decision on whether a new model should be trained. If a pre-trained model exists, the initiator can choose to use it directly or to enhance this model using more training instances. If this is a new adaptation task or the initiator chooses to enhance an existing model,

the master broadcasts the adaptation task to participating mobile devices with running worker software. The adaptation task description contains directions to the prospective workers. DeepCham supports two scenarios of adaptation tasks. In a first scenario, a task can ask mobile users to use existing photos in their devices. In a second scenario, it may ask them to take and label photos of the target domain within a finite time interval. If a user is willing to participate, it sends an acknowledgement back to the master and this device will be referred to as a worker. Once the workers have been identified, they start executing the task as a collaborative group coordinated by the master.

The initiator also specifies an expiration time for the adaptation task, and when the task expires, the master stops recruiting new workers and will train an adaptation model using training instances created by existing workers.

4.3.3 Domain-Aware Image Selection

The DeepCham worker client software running in each device scans the local storage to get a list of candidate images matching the task specification. The list of these candidate images can be very large and each image may contain multiple objects. Manually labeling a large number of objects is exhaustive and impractical. For example, if an image contains 3 objects on the average, and a user takes 5 seconds to label one object (i.e., create one training instance), labeling 100 images will incur 25 minutes! Therefore, DeepCham allows workers to specify a maximum number of images they intend to label. Given this maximum limit, DeepCham is supposed to select a subset of images which collectively contain a maximum number of distinct objects. A challenge that arises is that when multiple users are involved in the

adaptation training, the selection algorithm should also avoid having users label many duplicated objects. We solve this problem using a new distributed redundancy image pruning method.

Distributed Image Pruning

First, we describe a special simplest form of this distributed image pruning problem: there is only a single worker, u_i, who wants to label n_i ($n_i \in \mathbb{Z}^+$) images from the set of m_i (assuming $m_i > n_i, m_i \in \mathbb{Z}^+$) candidate images, and we would like to maximize the number of distinct objects covered in the selected images. However, we do not know what objects are included in each image and it is also not possible to identify them manually. To find a feasible solution, we observe that if the visual appearance of two images is considerably different, the possibility that the two images contain the same object is very small. Thus, instead of directly maximizing the number of distinct objects, we alternatively identify a subset of images with large visual difference. Our solution to the simplest single-worker scenario is straightforward: We assume that a global image feature (we use the lightweight and widely accepted CEDD [169] for evaluation) has been extracted from each candidate image, and we cluster the m_i candidate image features using *K-means* into n_i clusters. Then, for each cluster, we retain only one image which is the nearest neighbor to the cluster centroid.

Next, we describe formally a general version of this problem: For an adaptation task, there are l participating workers $\{u_1,...,u_l\}$ ($l \in \mathbb{Z}^+$). Each worker u_i ($1 \leq i \leq l$) wants to label a subset of n_i images from m_i candidate images in her local device, and she may join the task at any time point as long as the task has not expired.

We would like to maximize the number of distinct objects covered in the selected images of all workers. Again, we do not know what objects are included in each image. So we alternatively identify a subset of images with large visual difference.

For the general problem, simply running the clustering algorithm over extracted image features on each individual worker's device is not the optimal solution. The reason is that different workers may have taken similar photos, and without knowing what images are selected by other workers, duplicated objects may be included in the selected images of collaborating workers. Therefore, any solution must address this issue. Now, before we describe our algorithm to the general problem, we first consider two special scenarios:

One-after-another asynchronous scenario: In this scenario, we assume that a new worker, u_j, can only join the adaptation task after another participating worker, u_i, has selected the subset of images to retain. To make the new worker aware of what images are retained in other workers' devices, the system requires a worker to send the image features of the selected images to the master immediately after her image selection. The worker, u_1, who first joined this adaptation task, clusters m_1 candidate images to n_1 clusters without considering other workers' selection (as that in the simplest single-worker form). For each of the n_1 clusters, u_1 retains only one image which is the nearest neighbor to the cluster centroid, and then sends the image features of the retained images (or the cluster centroids for privacy consideration) to the master. A subsequent new worker n_i $(1 < i \leq l)$ first receives n_e image features of selected images by existing workers from the master. Then a 2-step method is executed on the new worker's device. In step 1, it clusters its m_i candidate images to $n_e + n_i$ clusters, and in each cluster it retains a single image which is the nearest

neighbor to the cluster centroid. The retained image features form the set $\hat{S}_r$. In step 2, we use each of the n_e image features received from the master as a query, and we find its nearest neighbor in $\hat{S}_r$ and remove it from $\hat{S}_r$. The remaining images in $\hat{S}_r$ are retained for the worker n_i to label, and it sends the image features (or centroids) to the master. To avoid incurring too much communication and computational cost, a predefined threshold $n_t (n_t \leq n_e)$ can be set by each worker so that the master will send at most n_t image features to the new worker. The set of n_t image features can be determined by either random selection or clustering over all n_e features.

All-together synchronized scenario: In this scenario, we assume that all workers join the adaptation task simultaneously, i.e., no single worker has selected a set of retained images before other workers, and this may happen when multiple workers actively accepting new adaptation tasks. Each worker u_i specifies the number of images she would like to label, n_i, and the master notify each worker that there are other existing workers. We propose a synchronized solution for this scenario: Each worker clusters her own m_i candidate images to $n_i + t$ clusters, and for each cluster, it retains the nearest neighbor to the cluster centroid. Here, we first select t more images for each worker in the expectation that we can find n_i images from the $n_i + t$ retained images that are unique to this worker. We name t as the synchronization allowance variable and discuss the setting of t in our experimental study. Each worker then sends the retained image features (or centroids) to the master. On receiving all the features/centroids, a second *K-means* algorithm is executed on the master to cluster them to $\sum n_i$ clusters. For each new cluster, if the cluster members are from two workers, just one of them is retained. The information (e.g., id) of the retained images is sent back to each worker.

Algorithm 6 Distributed Image Pruning (Worker Algorithm)

Input: n_i = the number of images to retain on the device
S_c = the set number of images features extracted from m_i candidate images
S_e = the set of n_e existing image features received from master
t = synchronization allowance variable
Output: Img_r is the set of retained images

```
1: function RemoveExisting(Ŝ_r, S_e)
2:     for each item in S_e do
3:         search the nearest neighbor f_nn of item in Ŝ_r
4:         remove f_nn from Ŝ_r
5:
6: function Pruning(S_c, S_e, n_i)
7:     if there are synchronized workers then
8:         Ŝ_r = clusterCentroid(S_c, n_i + n_e + t)
9:     else
10:        Ŝ_r = clusterCentroid(S_c, n_i + n_e)
11:    RemoveExisting(Ŝ_r, S_e)
12:    send Ŝ_r to master
13:    if there are synchronized workers then
14:        updateFromMaster(Ŝ_r)
15:    Img_r = ids of images that are nearest neighbors to centroids in Ŝ_r
16:    return Img_r
```

Solution to the general problem: The general problem can be solved by combining the solutions of the above two special scenarios. We present the pseudo-code in Algorithm 6 which is run on each worker. First, each worker would receive the set of existing features from the master, and it checks with the master if there are synchronized co-workers. If no co-worker exists, it falls back to the *one-after-another* asynchronous scenario. Otherwise, it first runs the asynchronous method but to get t more clusters and relies on the master for a second round clustering as that in the *all-together* synchronized scenario. The *updateFromMaster* function receives a list of identifiers for the retained images from the master.

4.3.4 Training Instance Generation Pipeline

On each participating mobile device, DeepCham worker software passes the selected images through the training instance generation pipeline. For an image containing

multiple objects, the pipeline first automatically detects objects within the image, and crops them out as training object images, i.e., an image contains exactly one single object to be labeled. This automation is achieved by the new designed *enhanced bounding boxes strategy* (EBBS) algorithm. EBBS operates on the output of a state-of-the-art bounding box proposal algorithm, *Edge Boxes* [165], to select a few object bounding boxes with high recall. (More on Section 4.3.4).

Figure 4.3: Interactive Labeling.

Once the training object images are created, they are input into the pre-installed generic deep-learning model which produces (1) a list of potential correct labels for each object image, and (2) the ConvNet features [39] extracted from one of its hidden layers. A *Worker* user selects the correct label by tapping on the screen (Figure 4.3). This design is based on our observation that though the recognition accuracy (Top-1) of a generic deep learning model is not sufficiently good ($< 60\%$ for the ILSVR2014 dataset), it is quite likely that the correct prediction is included in the Top-K results (around 90% for Top-5) [170]. Furthermore, since each user is asked

to label objects in his own photos, there is a better chance for him to know what the objects are in his own photos than labeling any object from others' photos. The simultaneously extracted ConvNet feature has been proved to be quite discriminative and has shown excellent performance in many different types of applications such as image retrieval [171] and scene recognition [39].

Each user selected label and the ConvNet feature pair forms a new training instance for the adaptation. For an object image which does not have its correct label presented in the suggested labels, there are 3 options: First, a user can manually input the correct label but use Autocomplete Suggestion as in Google Instant [172]. Second, if a user does not recognize the object, he can label it as a special class "other". Finally, a user is also given the option to skip labeling of this image.

Enhanced Bounding Boxes Strategy (EBBS)

Bounding box proposal in our training instance generation pipeline has different challenges and requirements from the traditional bounding box proposal algorithms used for known object detection tasks [166][165]:

- *No prior knowledge:* when an adaptation task is initialized, it is hard to know the set of available objects in that target visual domain.
- *Enough margins for cropped objects:* good training images for the adaptation task should have margins around the target objects to capture the target visual domain information [38].
- *Few bounding boxes with high recall:* the system requires users to select the correct label for each object/box and hence should identify fewer bounding boxes to minimize users' labeling efforts.

- *Overly small boxes should be ignored:* too small boxes are unsuitable for adaptation training due to low image quality.

EBBS meets the requirements and address the challenges by operating over the output of Edge Boxes algorithm [165], which is a list of ranked bounding boxes. Edge Boxes rank the boxes by calculating an objectiveness score which measures the number of edges in the box minus those are members of contours that overlap the boxs boundary. EBBS limits the maximum number of proposed bounding boxes from one image to a small value j (we set $j = 5$ in our experiment) and includes the following operations:

Removal of large and small boxes: Large bounding boxes proposed by EdgeBoxes will normally cover more than one object in a multi-object image. On the other hand, if an object covers a large area of the photo, it is typically in a one-object image, which does not require the bounding box identification process. Small bounding boxes typical contain unclear images and may not be good training instances. During our experiments, if the area of a proposed bounding box is larger than 40% of the image area, then we consider it to be "too large". If the area is smaller than 1% of the image area, we consider it to be "too small". We ignore all "large" and "small" boxes.

Equal regional distribution: We enforce equal regional distribution of bounding boxes in a given image as follows: First, we divide a given image into 4 equal regions: Upper Left, Lower Left, Upper Right, Lower Right. Next, we check if the bounding box we are considering, b_i, can be assigned to one of the 4 regions using the function Equal_Region in Algorithm **??**. This function ensures that (i) only a maximum of j bounding boxes are retained among all regions, (ii) the maximum number of boxes

assigned to a region is $ceil(j/4)$.

Add margins to bounding boxes: For a bounding box with width w and height h, we crop a larger box with width $1.5 * w$ and height $1.5 * h$ while keeping the original box center.

The pseudo-code for EBBS is illustrated in *Algorithm* 7. In our prototype Android implementation, we run Edge Boxes (and EBBS) on low resolution images (by resizing original images) for time efficiency. However, the bounding boxes returned from low-resolution images can be very small and the features extracted from such small patches are unreliable. Therefore, after we get the coordinates for the bounding boxes, we map them to the original high-resolution image and crop them out for the next process.

Algorithm 7 Enhanced Bounding Boxes Strategy

Input: L is the list of boxes in descending order of scores
Output: L_{top} is the list of the top 5 selected boxes

```
1: function SELECT(L)
2:     L_top ← ∅
3:     region_dist ← [0, 0, 0, 0]                    ▷ region distribution of L_top
4:     for each b ∈ L do
5:         if L_top.size ≥ j then
6:             break
7:         if b is too large || b is too small then
8:             continue
9:         if ¬ Equal_Region(b, region_dist) then
10:            continue
11:        Update(region_dist)
12:        L_top.add(b)
13:    Add_Margin(L_top)
14:    return L_top
```

Multi-scale Merge: We observe that when images are scaled to different resolutions, Edge Boxes will detect objects at different detail levels, and thus will produce different sets of bounding boxes using *Algorithm* 7. The union of each set of proposed bounding boxes produced using different image resolutions covers more complete objects. Unfortunately simply returning the union of boxes at different

scale may be an overkill and compromise our design goal of limiting the number of boxes for a user to label. Therefore, we propose a box merging method which removes redundant boxes from the union set. The redundant boxes are boxes with "nearly-the-same" visual content. In our prototype system, we define two boxes as having "nearly-the-same" visual content if their intersection area exceeds 50% of the area of each individual box. For such boxes, we merge by generating a union of these two boxes.

4.3.5 Adaptation pipeline: model construction

When all *Participating Workers* finish generating training instances, adaptation can be performed by using the training instances to create a new shallow model using traditional training methods, e.g., Softmax or SVM. The adaptation training can be done either at the *Master*, or collaboratively using a distributed learning method [173]. The adaptation model will be shared with all participating users. This adaptation model as well as the training instances are stored in the local edge server with visual domain attributes and can be shared with other users with the same adaptation requirement. New users may use the model directly or the adaptation model can be incrementally trained with new training instances.

4.4 DeepCham Object Recognition

DeepCham uses both the adapted shallow model and a deep model for object recognition (Figure 4.4). Specifically, we use the late-fusion method to generate the final recognition result by considering the following benefits:

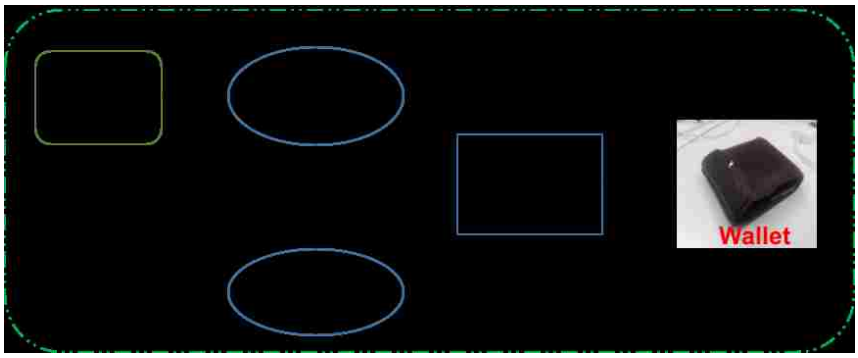


Figure 4.4: Object Recognition with Late-Fusion.

Good performance with a few training instances: It has been shown that the late-fusion method [38] achieves similar recognition accuracy improvement to a complicated adaptation method with one-shot adaptation, i.e., using a single training instance for each object class.

Simple adaptation training: To train a shallow learning model does not require a powerful GPU-equipped server. It is fast and can be performed even on a single mobile device.

Sharing adaptation as a plug-in: A shallow model has much smaller size than a deep model. Once it is trained, it can be easily shared with other mobile users who are interested in object recognition in such target visual domain.

The issue to apply the late-fusion method directly in mobile visual domain adaptation is that: it assumes that both the deep model and the shallow model recognize the same set of object classes. However, the set of object classes available in each target visual domain is highly variable and it is impractical to train or fine-tune a deep model for each domain. We solve this problem by proposing a domain-constrained deep model.

4.4.1 Domain-Constrained Deep Model

The domain-constrained deep model, created from the generic deep model, recognizes only the object classes included in a target visual domain. We construct the domain-constrained deep model as follows: let us assume that there are d object classes in a particular mobile visual domain (i.e., classes in the adaptation model). For each image, given the prediction vector obtained using a generic deep model, which represents the probabilities of that image belonging to different recognizable classes, we retain only those probabilities for the set of d classes $\{p_1, p_2, ..., p_d\}$. Then we normalize each probability $p_i = p_i / \sum_{n=1}^{d} p_n$ as the domain-constrained prediction result.

4.4.2 Late-fusion based Recognition

To obtain the final recognition result, we apply the late-fusion method [38][155][156] where a linear interpolation is used to compute a fused probability of a particular instance: $p_{fusion} = \alpha p_{deep} + (1 - \alpha) p_{adpt}$. (where p_{deep} is the probability from the domain-constrained deep model, and p_{adpt} is the probability from the shallow adaptation model). One issue here is that it is hard to determine the weighting factor α for the dual-model recognition. We provide a basic guideline for setting the weight in our experimental study.

The adaptation model may not include a complete set of object classes in a target visual domain especially when there is only a single worker involved for the adaptation training. We solve this problem by including a special "other" class in the adaptation model which represents unrecognizable during the Training Instance Creation step. When an object is recognized as "other by the adaptation model,

the system uses only the generic deep model for object recognition.

4.5 Evaluation

We first describe a popularly used CNN model which is referred to as the generic deep model (section 2.5.1), and a new image dataset we created capturing different target visual domains (section 2.5.2). Next, we use this dataset to present a baseline adaptation performance, i.e., we assume perfect training instances for an inclusive object set, and compare it with that achieved with the generic deep model (section 2.5.3). Then, we show the impact of our proposed distributed image pruning algorithm (section 2.5.4) and the EBBS algorithm (section 2.5.5), and the adaptation performance using training instances created by the proposed algorithms (section 2.5.6). Finally, we describe the prototype implementation and measure its system cost (section 2.5.7).

4.5.1 AlexNet CNN Model

We use the 1000-class AlexNet [29] CNN model, which is trained using the ILSVRC 2012 dataset. This model achieves 57.4% Top-1 accuracy and 80.4% Top-5 accuracy on the ImageNet 2012 validation dataset. AlexNet consists of 5 convolutional layers (Conv1-Conv5), some of which are followed by max-pooling layers, 3 fully-connected layers (FC6, FC7, FC8), and a final 1000-way softmax layer. In the experiment, we train the adaptation models using the 4096 dimensional ConvNet features extracted from FC7 [38].

Attribute	Values
Location	Home Apartment (home), Research Building (lab), Library (lib)
Time of Day	Noon (d), Evening (e)
Device/Resolution	iPhone 4s/3264*2448 (ip4), iPad Mini 2/2592*1944 (ipm), LG G2/4160*3120 (lg2)

Table 4.1: Meta-Data Attribute Values of MCID

4.5.2 Mobile Context Image Dataset (MCID)

To evaluate the performance of our system, we collected a new Mobile Context Image Dataset (MCID) which includes photos taken from 3 locations at two different times (noon/evening) using three different mobile devices. All photos are taken at the default resolution under the auto-focus and auto-flash (if flash is available) mode. Table 4.1 summarizes the meta-data image attributes included in our MCID dataset.

We can divide images in MCID into 18 subsets based on different combinations of device-type, location and the time when an image is captured. We represent each subset using the following format *dev_loc_time*, e.g., ip4_lab_d means images taken using iPhone4s(ip4) at a Research Building (lab) at noon (d). For subsets that are taken at the same location, their images contain nearly the same visual contents: each image in one subset must have mirror images in other subsets at that location (an example in Figure 4.5). This is arranged so that it is easier to determine which attribute has more impact on the adaptation performance. MCID has 117 images for each subset associated with the *Home Apartment*, 204 images at a *Research Building*, and 99 images at the library *Library*. Thus, the total number of images contained in MCID is (117+204+99)*2 (time)*3 (device)=2520. Each image may contain one or multiple objects.

To ease the discussion of our evaluation results, we first define three important

Term	Meaning
Class	The classification labels, such as Laptop, Cellphone.
Object	A particular object, such as my cellphone, Tom's cellphone.
Instance	An image/image patch showing a single object.

Table 4.2: MCID Terminologies

Location	Instance Num	Object Num	Class Num
Home Apartment	192	64	54
Research Building	231	77	39
Library	162	54	34

Table 4.3: MCID Statistics

terminologies shown in Table 4.2. We manually cropped instances from all images in MCID for our baseline evaluation. For each instance, it is annotated with (1) the image id from which it is cropped, (2) the coordinates for its bounding box in the original image, (3) the object label, e.g., Tom's cellphone, and (4) the class label. All class labels of the manually cropped instances in MCID are included in the 1000 classes of AlexNet. Each object to be recognized has exactly 3 instances showing its visual appearances from 3 different angles. Table 4.3 summarizes the number of *instances*, *objects* and *classes* for subsets at each location.

4.5.3 Baseline Adaptation Performance

In this subsection, we first compare the baseline results obtained using the generic Alexnet deep model and our proposed adaptation model using our manually cropped boxes, i.e., assuming bounding boxes containing all objects in the images within MCID are correctly identified.

(a) An image in ip4_lib_d subset. (b) The "mirror" in ipm_lib_e subset.

Figure 4.5: Example of mirror images.

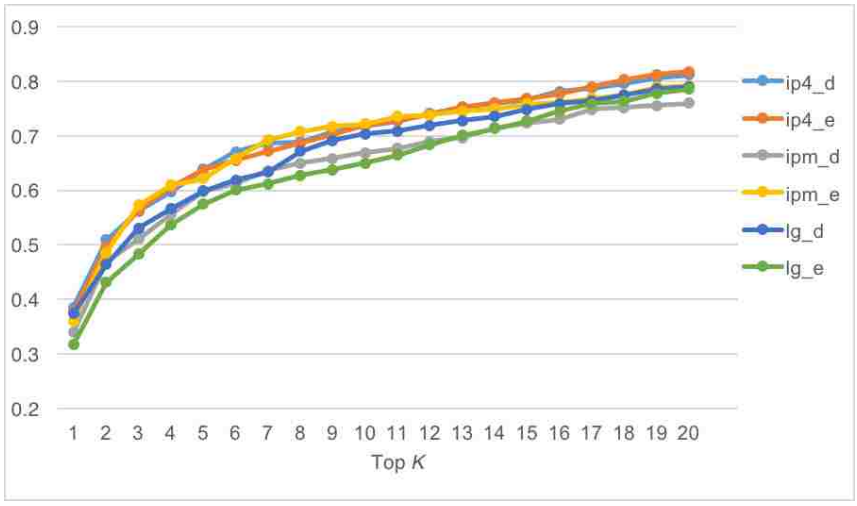


Figure 4.6: Generic Deep Model Performance.

Generic Deep Model Only Performance

We run an experiment to show the classification performance using Alexnet deep model for manually cropped object instances, and our result is shown in Figure 4.6. Each curve in the figure represents the average classification accuracy for a particular device/time combination at all 3 available locations.

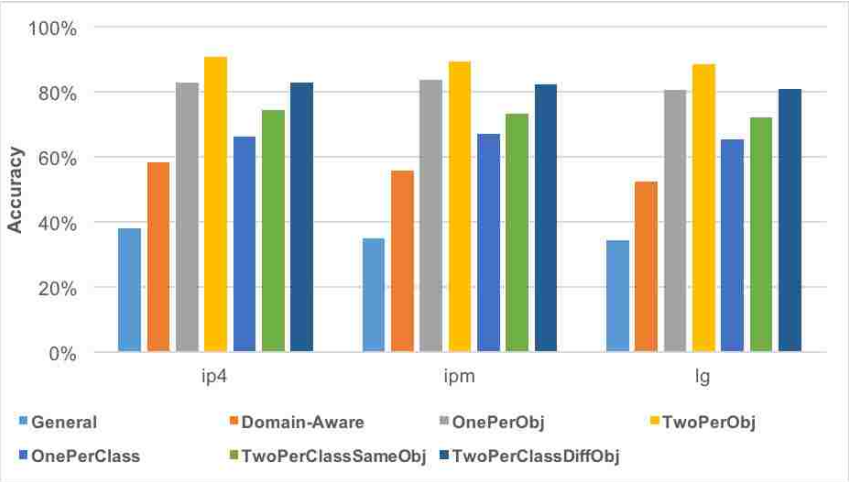


Figure 4.7: Adaptation Performance.

The purpose of this experiment is two-fold. First, we want to verify that using a generic deep model does not yield good *Top-1* classification result. The result presented in the figure confirms our hypothesis since the obtained *Top-1* accuracy is always below 40% for all device/time combinations, which means that a generic deep model is not practically useful in the mobile object recognition task. Second, we want to verify if the generic deep model achieves higher accuracy when K is large. For $K > 1$, we assume that as long as the ground-truth class label is in the Top-K list, we will consider the returned result to be correct. From Figure 4.6, we see that the classification accuracy improves to around 60% when $K = 5$ and 70% when $K = 10$. This result suggests that we can use this generic deep model to provide a useful suggested list to a DeepCham user during the training instance creation pipeline since it means he only has to manually input object labels 30% of the time if $K = 10$ is chosen.

Shallow Adaptation Model Performance

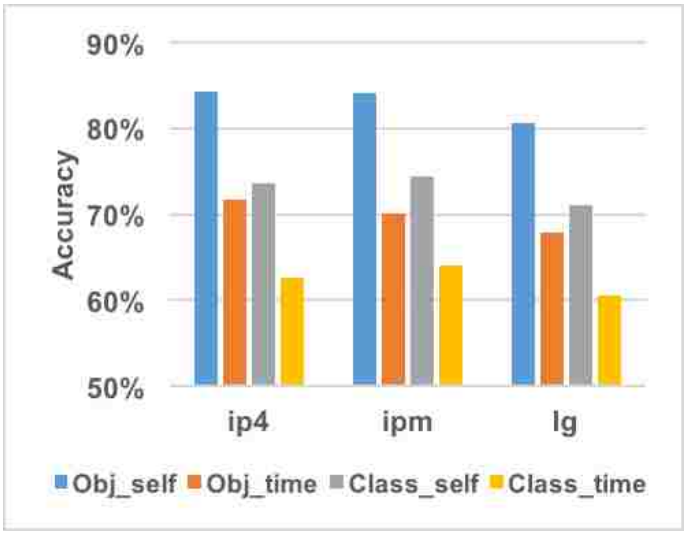


Figure 4.8: Time Variance.

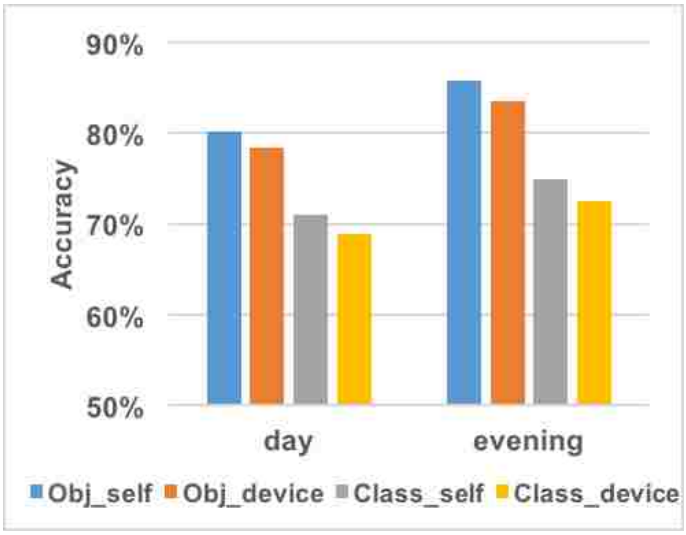


Figure 4.9: Device Variation.

Figure 4.7 shows the classification result using the adaptation model trained only using object instances within a subset. Each object in a subset, e.g., ip4_home_d,

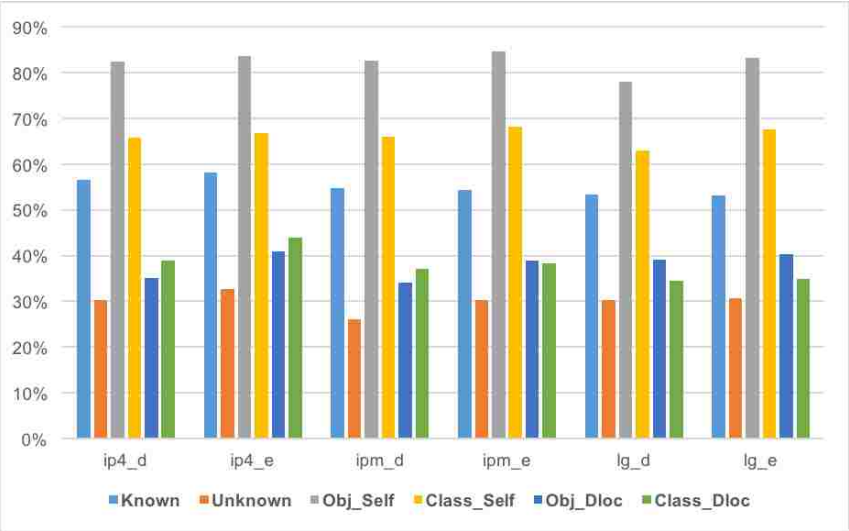


Figure 4.10: Location Variance.

contains three instances, thus we split each subset into 3 mini-sets where each miniset contains only one instance of any object. In the experiment, we guarantee that the adaptation model is trained and tested with different mini-sets. Seven different experiment settings are presented.

- *General* and *Domain-constrained*: *Top-1* accuracy using the generic deep model or the domain-constrained deep model (as described in 4.1). All images are used as testing images.
- *OnePerObj* and *TwoPerObj*: *Top-1* accuracy using the adaptation model trained using one or two instances for each object. One miniset is used for training while the other two are used for testing.
- *OnePerClass*: *Top-1* accuracy using the adaptation model trained using one instance for each class. One miniset is used for training while the other two are used for testing.

- *TwoPerSameObj* and *TwoPerDiffObj*: *Top-1* accuracy using the adaptation model trained using two instances for each class. These two selected instances are either representing the same object (the former case) or different object (the latter case). Two minisets are used for training while the 3rd is used for testing.

Each bar in the figure shows the average accuracy for a single device using 6 subsets (3 locations and 2 different time-of-day). For the presented result, we observe that: (1) Domain-constrained model yields a significant increase of 20% in the *Top-1* accuracy compares to the general model. (2) Classification using the newly trained adaptation model always has better accuracy than that achieved using either the generic or domain-constrained models. Adaptation with one training instance per object (82.4%) improves accuracy by 46.6% (compared to general model), and 27.0% (compared to domain-constrained model). (3) Training using instances covering more objects within a class is better than training with more instances representing the same object.

Impact of varying image attributes

We study the impact of different attributes on the accuracy achieved by the adaptation model. We run experiments using different subsets with varying attributes for training and testing.

We first present the results on the impact of the time attribute in Figure 4.8. For this experiment, we choose image subsets with same location and device type but different time attribute for training and testing, e.g., ip4_home_d and ip4_home_e. Again, we make sure that the training instances and testing instances are cropped

images of the same object taken from different angles. Here we use 4 different experimental settings: *Obj_self* and *Class_self* represent the *Top-1* accuracy with adaptation model that is trained using one instance per object or one instance per class, while the testing instances are from the same subset as the training instances. *Obj_time* and *Class_time* represent the *Top-1* accuracy with adaptation model that is trained using one instance per object or one instance per class, while the testing instances are from a different subset (time attribute).

Similarly, we present results showing the impact of device type in Figure 4.9. The *Obj_device* and *Class_device* represent the *Top-1* accuracy with the adaptation model trained using one instance per object or one object per class, but the testing instances are from another subset with different device type but at the same location and time.

To evaluate the impact of location attribute, we identify 21 common classes among objects contained in the photos for the 3 locations in MCID. We run multiple experiments as follows: in each experiment, we train a 21-class adaptation model using one subset of photos from one location with the same device type and time attributes and test this classifier using appropriate subsets from the other two locations. The experiment results are shown in Figure 4.10. The *Obj_Dloc* and *Class_Dloc* represent the *Top-1* accuracy with adaptation model which is trained using one instance per object or one object per class, but the testing instances are from another subset with different locations.

Comparing the results presented in Figure 4.8, Figure 4.9 and Figure 4.10, it is clear that the location attribute has the most significant impact on the adaptation performance. An adaptation model trained at a different location is only slightly

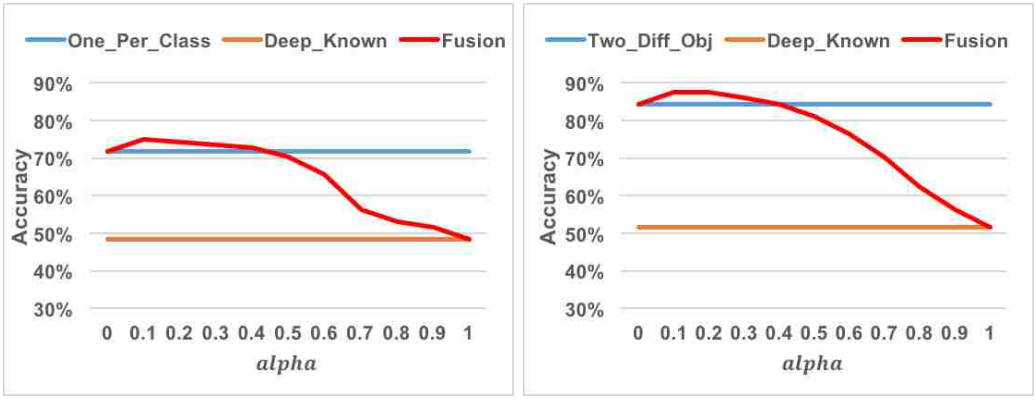
better than the generic deep model. On the other hand, the time attribute has greater impact on the performance than the device type. Training using images under different time attribute makes the accuracy drop more than 10%, while training with images taken by a different device has negligible impact. This result assures the potential benefit of collaborative model training involving many different types of mobile devices.

Late-Fusion Study

We study the power of late fusion and present the results in Figure 4.11 for two subsets, ip4_home_d and ip4_lab_d. The subset ip4_home_d includes fewer number of objects for each class (1.19) than the subset ip4_lab_d (1.97). We train the shallow adaptation model using either one randomly chosen training instance per class (One_Per_Class) or two randomly chosen training instances per class (Two_Per_Class). We observe from the result that: while the accuracy difference between two models is small (Figure 4.11(c)), the fusion method always achieves better performance with up to 10% accuracy improvement. When the accuracy difference between two models is large (Figure 4.11(a), 4.11(b), and 4.11(d)), the benefit of using late-fusion is very limited. Furthermore, our results show that when $\alpha < 0.4$, the late fusion method will always improve the accuracy of the better model (i.e., the adaption method).

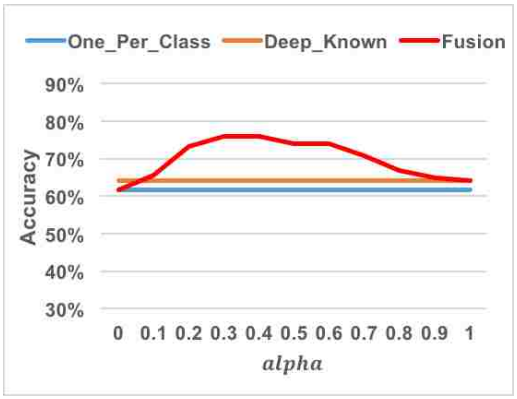
4.5.4 Impact of Distributed Image Pruning

To evaluate the effectiveness of our proposed distributed image pruning algorithm, we define the *object coverage rate* (OCR) for all involved workers as our performance

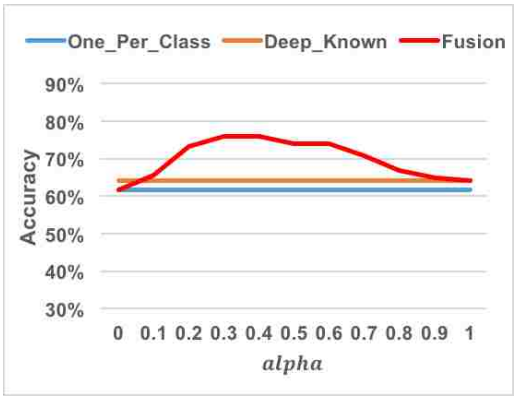


(a) One_per_class (ip4_home_d).

(b) Two_per_class (ip4_home_d).



(c) One_per_class (ip4_lab_d).

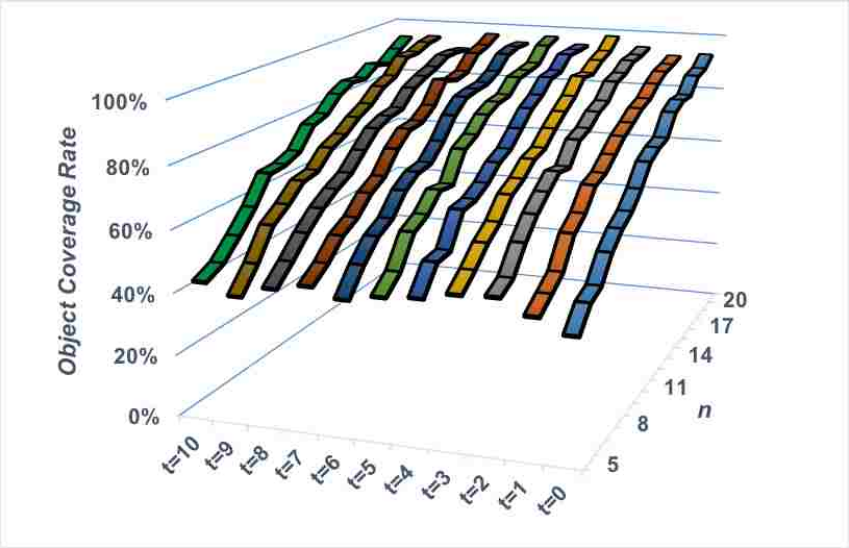


(d) Two_per_class (ip4_lab_d).

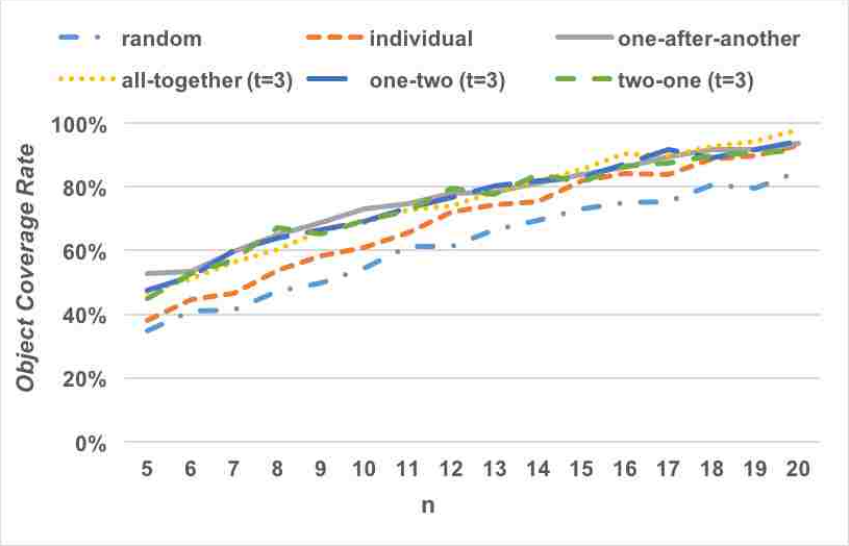
Figure 4.11: Late-Fusion Performance.

metric. OCR is calculated as follows: assume that the m_i candidate images for worker u_i contain a set of M_i objects, then we generate $S = \bigcup M_i$ as a union set from all workers. Next, assume that the set of identified objects from the n_i retained images is R_i. We also generate $R = \bigcup R_i$. Next, the object coverage ratio is computed as $\frac{|R|}{|S|}$.

To evaluate the performance of our proposed distributed image pruning algorithm, we assume that there are 3 synchronized workers a, b, and c. For each



(a) t: The Synchronization Allowance Variable.



(b) Performance Comparison.

Figure 4.12: Distributed Image Pruning Performance.

worker, we randomly choose 50% of 351 images taken at home_day target domain as the candidate images to evaluate in our experiment. Thus, each worker has roughly 175 images, and the likelihood of each user having images containing the same object is relatively high. In our experiment, we assume that each worker will label the same number of images, i.e., $n_a = n_b = n_c = n$ ($5 \leq n \leq 20$). We repeat all experiments 20 times.

We first study the impact of varying t, the synchronization allowance variable, on the performance of our proposed algorithm. We run the experiment using a scenario with 3 workers are all synchronized at the master, i.e., the workers do not receive existing features from the master and they all cluster their candidate images to $n + t$ clusters. The result of this experiment is presented in Figure 4.12(a). It shows that when $t \leq 3$, the performance of our algorithm increases as t grows; when $t \geq 3$, the performance is relatively stable and is not sensitive to the setting of t.

Next, we show the performance of the proposed algorithm in 4 different scenarios: *one-after-another* simulates the one-after-another asynchronous scenario and a new worker can only join the adaptation task after an existing worker has finished the image pruning selection; *all-together* simulates the all-together synchronized scenario and all workers are synchronized at the master; *one-two* simulates the combined scenario that one worker joined the task first and the remaining two workers joined the task together after the first worker finished the pruning selection; finally, *two-one* simulates the combined scenario where two workers joined the task together initially, and the third worker joined after these two finished the pruning selection. In all scenarios involving synchronized workers, we set $t = 3$. We also compare with two simple non-distributed methods: *random* means that each worker randomly

select a subset of n images; *individual* means that each worker runs the clustering locally without cooperating with others. We present the experiment result in Figure 4.12(b).

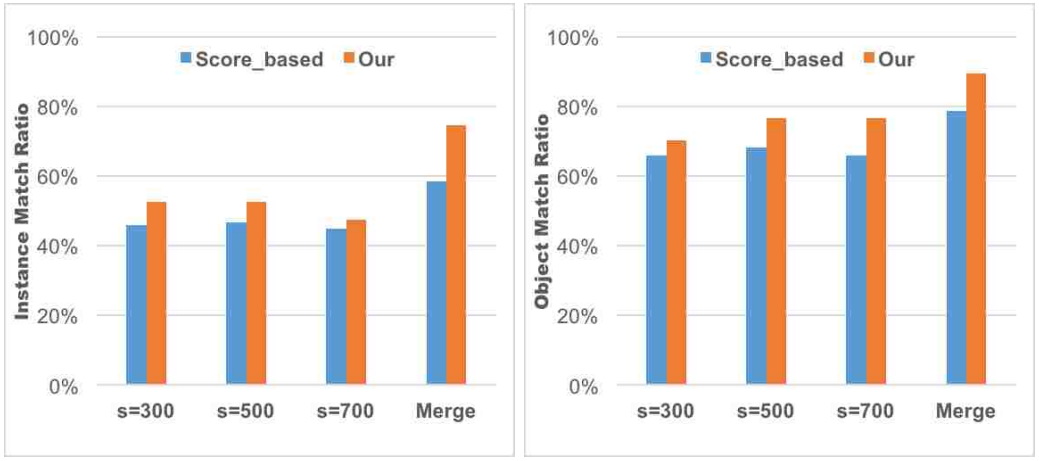
The performance result shows that compared to the simple non-distributed methods, our proposed algorithm improves the OCR in all experiment scenarios, especially when n is small. For example when $n = 7$, the average OCR of our algorithm in the four experiment scenarios is 58.3%, while the *random* method is 41.5% and the *individual* method is 46.5%. This result is very promising since asking each user to label fewer images will increase their incentives to join the task. On the other hand, when n is very large, e.g., $n > 15$, the OCR of the *individual* method is getting close to the proposed algorithm (both are approaching 100% OCR). However, in practice, a user is not likely to label a large number of images.

4.5.5 EBBS performance

We evaluate our enhanced bounding boxes strategy (EBBS) using 59 multi-object images in ip4_home_d. These images include 118 instances of 47 objects that are recognizable by the Alexnet model. For each bounding box proposed by EBBS, if it overlaps 80% of a manually cropped box and its area is less than 1.5 times the manually cropped box, then we consider this as a **correct match**. We use the default parameters of the original EdgeBoxes algorithm.

In this set of experiments, we run EBBS over 3 different image scales (300, 500, 700) [1] and we also evaluate our merge method over the 3 scales. For each single scale, EBBS returns 5 bounding boxes for a input image. We measure the performance

[1] Resizing an image to scale s means that we keep the original aspect ratio and rescale the longer edge to s



(a) Instance Match Ratio. (b) Object Match Ratio.

Figure 4.13: Bounding Box Selection Performance.

using two metrics. First, the *instance match ratio* (IMR): let k_i be the number of correct matches within image i, then IMR is computed as $\sum k_i/118$. the *object match ratio* (OMR): we count the total number of "recognizable" objects contained in all the correct matches in these 59 images (denoted as w) and compute OMR as $w/47$.

In Figure 4.13, we present our results which compare our method with a simple score-based selection method (top 5 scored boxes). Figure 4.13(a) shows the *instance match ratio* (IMR) while Figure 4.13(b) shows the *object match ratio* (OMR). The results demonstrate that our strategy is more effective in including more potential objects than the score-based selection method. Furthermore, our proposed multi-scale merging method improves both IMR and OMR significantly compared with the single-scale results. In addition, after merging, on the average, the number of selected bounding boxes is about 7 which is still acceptable.

4.5.6 Object Recognition Accuracy using DeepCham Generated Training Instances

We evaluate the object recognition accuracy in the home_d target visual domain using training instances generated by DeepCham. The training instances is generated as follows. First, we assume there are 3 synchronized workers, each who wants to label 10 images, and the candidate images of each worker are the same as that used in section 4.5.4. Then, we run the distributed image pruning to select the 30 images for the 3 workers. Among the 30 images, there are 22 multi-object images and 8 single-object images, and we run EBBS with 3-scale merge on the 22 multi-object images to propose object bounding boxes.

The following images/boxes are used to produce training instances: (1) the 8 single-object images and (2) the proposed bounding boxes each of which is a correct match and its correct label is in the Top-10 list produced by the AlexNet model. In total, we have 36 training instances for 27 classes, and we use these instances to train an adaptation model. The recognition accuracy of the adaptation is tested on manually cropped bounding boxes containing objects belonging to the same 27 classes from the home_d target visual domain. The top-1 accuracy with our late fusion method ($\alpha = 0.3$) is 82.7%. On the other hand, the top-1 accuracy using the generic AlexNet model is only 33.4%. Thus, we achieve around 150% accuracy improvement.

4.5.7 Prototype System Evaluation

Prototype Implementation

We implement our preliminary prototype system with a Linux server as the master edge server and Android mobile devices as workers. The Linux server and a Wi-Fi router is connected using a 1 Gbps Ethernet switch, and the Android devices communicate with the Linux server through the Wi-Fi router. On each Android based worker, we install the Alexnet CNN module within the Android Caffe [150] framework which contains both the trained CNN model (included in the Caffe Model Zoo [174]) and libraries that support deep CNN feature extraction. In addition, we re-implemented the EdgeBoxes algorithm from Matlab [175] to OpenCV environment, and then ported the re-implemented module using JNI to run on Android devices. As far as we know, this is the first working implementation of EdgeBoxes algorithm in OpenCV as well as on Android platform. We also use LibSVM [176] to train the shallow model. Furthermore, we use the K-means++ [177] algorithm for fast convergence and set the maximum number of iterations to 100.

Prototype Evaluation

We focus our evaluation of the preliminary prototype system on the Android worker. Specifically, we use an Android Device LG G2 which has a Snapdragon 800 chip with Quad-core CPU @ 2.26 GHz and 2 GB RAM. We measure the system component running time, the memory cost and the energy cost.

System Component Running Time: The running times for each more computationally expensive step during the adapted training (AT) and recognition (R) process of our initial prototype system are summarized in Table 4.4.

Stage	Component	Time (ms)
AT-0	CEDD	214
AT-1	K-Means (40 clusters of 200)	897
AT-2	EBBS-300	357
	EBBS-500	792
	EBBS-700	1220
	EBBS-Merge	1459
R	AlexNet-Classify	697
	SVM-Classify	85
	Classify-Total (Late-Fusion)	793

Table 4.4: System Running Time

Row 1 shows the cost of CEDD feature extraction which occurs when a photo is taken. The result shows that this phone can extract CEDD features from 50 images in just 10 seconds. Row 2 shows the cost of K-means clustering and it takes 897ms to cluster 200 CEDD features to 40 clusters. Next 4 rows show the running time of EBBS with images of different scales. The algorithm runs faster on small-scale images and slower on large-scale images. The "EBBS-Merge" row shows the running time with a parallel computing scheme (i.e., we run the algorithm for each image scale using a different thread). With multithreading, it costs less than 1.5 seconds per image to run our Enhanced EdgeBox selection scheme and hence should not produce user-observable delay.

During the querying process, both the AlexNet deep model and the SVM-based adapted model will be used before late-fusion method is applied. Thus, the next 3 rows show the cost of classification using the deep or adapted model, and the overall classification cost. Our results show that the extra cost incurred using the shallow adapted model (85ms) is small compared to the cost using just the deep model (697ms).

System Memory Cost: The memory cost for DeepCham is summarized in Table 4.5. During the adaptation training phase, the worker loads a trained *Edge*

Model within EdgeBoxes, and the deep *AlexNet Model* for training instance generation. The total memory cost for training is 238.66 MB. In the object recognition phase, the system loads the *AlexNet Model*, and the context-aware adaptation model, with a total memory cost of 234.95 MB. Considering the large RAM equipped on modern mobile devices (e.g., 2GB on LG G2), such memory cost is acceptable. Note that the main memory cost is due to the large deep model, and the remaining parts of the system has minimum memory impact. On the other hand, there has been techniques proposed to reduce the memory cost of the deep learning model such as [178], we will consider such optimization in our future work.

Stage	**Component**	**Memory (MB)**
Training	EdgeModel (EdgeBoxes)	6.09
	AlexNet	232.57
	Total	238.66
Recognition	AlexNet	232.57
	SVM	2.38
	Total	234.95

Table 4.5: System Memory Cost

Energy Consumption: Another concern on the adaptation training is the energy consumption on each participating worker device. DeepCham saves processing cost in two ways. First, it uses the clustering algorithm to select a representative subset of images (which is controlled by the participating users) to process. Second, DeepCham achieves energy saving by processing images locally instead of sending images from a mobile device to a local Edge Server. In our energy measurement using PowerTutor [179], we assume that a user has 100 candidate images matching the target visual domain, 10 out the the 100 images are selected by the image pruning algorithm, and there are 50 objects (used to create training instances) identified from the 10 images. We measure the energy cost (CPU+WiFi) for each component of our

system and present the result in Table 4.6[2]. The total energy consumption is only 59.1 joules which is only 0.14% of the LG G2's battery capacity. Meanwhile, this energy consumption is only 85.7% for that consumed to read 10 minutes of Facebook News Feed, and 35.6% for that consumed to watch 10 minutes of YouTube.

	Component	**Energy Cost (J)**
DeepCham	CEDD feature (100 images)	23.1
	Image pruning (10 of 100)	0.3
	Edge Boxes & EBBS (10 images)	9.8
	Transfer training instances (50 instances)	5.9
	ConvNet feature (50 object images)	20.0
	Adaptation Training Total	59.1
Comparison	Facebook News Feed (10 Mins)	68.9
	YouTube (10 Mins)	165.8

Table 4.6: Energy Consumption For Adaptation Training (CPU+WiFi)

We compare the energy cost to two other systems:

Purely Server-side Image Processing: All images matching the target visual domain are sent to a server for processing. The server sends back the bounding box coordinates and the suggested labels to the client. Since our system works on high resolution images, and on average each image is around 2MB, the purely server-side image processing system cost around 400 joules (6.76 times of DeepCham) battery energy mainly due to sending 100 images to the server.

Combined Server-Client Image Processing: The mobile client runs the image pruning algorithm locally and sends the selected images to the server. The server sends back the bounding box coordinates and the suggested labels to the client. The energy cost for such system is around 63.3 joules, and DeepCham saves around 9% energy.

[2]Note that there is also energy consumption for LCD display in the interactive labeling process of adaptation training (as well as Facebook and YouTube apps), and this may vary due to different user settings.

In addition, DeepCham protects user privacy since no raw images are sent out of each participating user's device which is a feature neither of the compared systems contain.

4.6 Discussion

How to incentivize users for participating in the system?

For adaptation tasks in private or business spaces, the training can be performed by companies whose employees perform a per-site object training. Examples include entertainment venues, real estate services, etc. For adaptation tasks in public spaces (e.g., in a museum), the task initiator (or organizer) may give rewards (either physical or virtual) to participants. For home environments, other incentives may also exist, e.g. users of a mobile robot are inherently motivated to collaboratively create higher accuracy models that the robot can use to increase its intelligence.

How often do we need to retrain the system based on domain specific changes?

Each adaptation task is specific to a particular target visual domain. A change in visual domain (e.g., to recognize objects in rainy day instead of sunny day) can trigger a new adaptation task (not re-training). For target visual domains that have already been adapted, if there are minor changes (e.g., new objects added), the system may incrementally train the shallow adaptation model (e.g., SVM) [180] by adding new instances for those previously uncaptured objects.

What if some users inject false labels into the system?

In real life, some users may be malicious and deliberately label the objects wrongly.

Existing defense mechanisms [181], e.g. letting more users to label the same object and use the majority vote wins strategy to select the right label, can be used to overcome such malicious attacks. We leave such evaluation of the defense mechanism for future work.

Can we select the later fusion parameter dynamically?

In this chapter, we do not explore the dynamic hyper-parameter tuning problem for late fusion. The reason is that when the number of labeled training/validation is small as in our proposed adaptation process, existing hyper-parameter tuning methods [182, 183] may not be able to be applied directly. However, it may be feasible to monitor the inference error rate of the generic deep learning model in the interactive labeling process and set the hyper-parameter α accordingly (in equation of section 4.2), i.e., if the generic model gives bad suggestion for labeling of the adaptation training images (e.g., correct labels are ranked low), we may set a small value, and vice versa. We hope to explore this in our future work.

In scenarios of the adaptation training campaign, can we just ask participants to take a diverse set of photos instead of running the distributed image pruning algorithm?

This is a trade-off. To avoid running the distributed image pruning algorithm, recruited participants must be instructed to take different subsets of object related photos. Such an approach creates more burden to participants as well as task initiators and hence less attractive compared to the distributed image pruning solution.

4.7 Summary

In this chapter, we have presented DeepCham, an edge-mediated collaborative deep model adaptation framework for mobile object recognition. DeepCham is the first work that addresses the challenge one faces in deploying a generic deep learning model in the highly variable mobile visual domains. DeepCham learns a domain-specific shallow model with training instances interactively created by collaborative users using local images in their mobile devices. DeepCham relies on a distributed redundancy image pruning algorithm, an enhanced bounding box proposal algorithm as well as a generic deep model to facilitate the creation of training instances. Experimental studies shows that DeepCham enables efficient and effective adaptation training, and its object recognition accuracy significantly outperforms that achieved using a generic deep model.

Chapter 5

Conclusion

So far we have presented several robust mobile visual recognition systems, i.e., EMOD for efficient mobile image retrieval with the bag-of-visual-word framework, MOCA for energy-efficient and privacy-preserving deep processing of mobile vision apps, and DeepCham for deep learning model adaptation of mobile visual domains, from Chapter 2 to Chapter 4. In this chapter, we conclude this dissertation. We first summarize our research findings and contributions, and we also discuss future research directions.

5.1 Summary

Automatically recognizing and understanding visual content captured by mobile devices' cameras has become a particularly important feature for many mobile applications. However, it is non-trivial to build a robust mobile visual recognition system due to many challenges we have identified and discussed, e.g., constrained computing power, storage space and battery capacity on mobile devices, privacy

concern using a mobile-cloud architecture. We address those challenges with an overall system design approach which is guided by mobile context understanding, privacy awareness and algorithmic optimization and involves heterogeneous computing resources, e.g., mobile devices, edge server as well as the cloud.

In Chapter 2, we proposed EMOD that enables efficient on-device image retrieval on a context-aware partial image database. EMOD optimizes the bag-of-visual-word (BOVW) image retrieval framework with algorithmic improvement on each component of the processing pipeline. In addition, we proposed a two-stage k-NN based re-ranking method that improves the retrieval accuracy dramatically with minimal overhead. Further response time reduction is achieved by exploiting the parallel computing capabilities made possible by multi-core mobile processors. Our EMOD prototype system on Android platform achieves real-time high-accuracy retrieval with significantly reduced memory and energy consumption.

Deep learning based techniques have greatly advanced the visual recognition capability of machines. However, running such computational intensive algorithms on resource-constrained mobile devices is still daunting. On the other hand, a cloud-based solution reduces the burdens on mobiles, but at the risk of leaking sensitive user data to untrusted cloud servers. In Chapter 3, we proposed MOCA that decouples the hierarchical neural network layers across mobile and cloud for energy-efficient and privacy-preserving mobile deep visual processing. MOCA leverages the privacy-preserving feature maps extracted from deep models to protect personal data privacy. MOCA also enables differentiated privacy settings via a DNN-based privacy quantification framework to meet the varying privacy requirements of different mobile applications. To further accelerate the feature extraction procedure

and reduce the feature size on mobile devices, several optimizations are proposed and implemented. Experimental results with a variety of DNN-based mobile applications on several large-scale datasets demonstrate that the privacy of these data are well preserved, while the energy overhead on mobile device is up to 6.7x lower than the state-of-the-art approach.

In Chapter 4, we proposed DeepCham that adapts trained generic deep learning models to target mobile visual domains to address the domain bias problem. DeepCham leverages abundant metadata-rich (e.g., GPS location, time) photos that have already been taken by mobile users, and trains a shallow adaptation model (e.g., SVM) in real-time on a mobile device. To achieve this, DeepCham includes an attribute-based visual domain specification module that identify photos matching a target visual domain, an efficient object bounding box proposal method to locate objects shown in a photo, and an interactive object labeling pipeline which suggests labels for an input image using a generic deep learning model. Similar to Moca, user privacy is well preserved since only features extracted from user photos may leave a mobile device. DeepCham obtains 2x accuracy improvement on a new collected Mobile Context Image Dataset (MCID) and is demonstrated to be more energy-efficient than cloud based solutions.

Through this dissertation, readers learned characteristics, properties, challenges in the design of robust mobile visual recognition systems, especially in bag-of-visual-word based image retrieval systems and deep learning based systems. We also expect that readers could acquire knowledge of algorithmic optimization of popular visual recognition methods. In addition, we hope readers can be inspired by some methodologies we proposed such as mobile context understanding and employing

heterogeneous computing resources, and can apply them to other application domains.

5.2 Future Work

Based on this thesis, there are many possible extensions of the current approaches. In this section, we suggest a few future research directions.

Smart Vision Algorithm Selection. One computer vision task may be accomplished by various types of algorithms each has a different accuracy, computing complexity and energy impact on the mobile devices. For example of object recognition, some objects may only be correctly recognized by a computing-intensive deep neural network with more than 100 layers, while some other can be easily recognized by a much smaller neural network with a few layers that can efficiently run on a mobile device. Therefore, it would be interesting to explore how a smart algorithm can be devised to automatically decide which vision algorithm should be used to guarantee recognition accuracy while minimizing the energy cost.

Personalized Mobile Context Understanding. Understanding the context of a particular mobile user has many benefits including reduced recognition space and better recognition result. For example, knowing a user's favorite brand can guide a product search app tend to promote rankings of this brand. To infer such knowledge, multiple sources of mobile data can be easily obtained such as a user's search and location history, mobile sensory data such as gyroscope, accelerometer. An interesting research question is: How should a visual recognition engine/model built by summarizing various types of data input?

Heterogeneous mobile processing units. Mobile chip makers are developing novel mobile processors with heterogeneous processing units other than CPU such as mobile GPUs, ARM NEON supported mobile co-processors, the Neural Processing Unit (NPU) in Qualcomm's Zeroth Platform. By exploring the characteristics of different processing units, further optimization such as better energy efficiency may be achieved by decoupling the computation and deploy each piece to the most appropriate unit.

Make the most of the Edge. With the development of internet of things (IoT), edge computing units (or cloudlets) have been deployed with considerable computing and storage power. Such edge units may assume part of the functionality of the cloud , but have faster response (in terms of round-trip latency) and better privacy promise. For example, Nvidia's Jetson TX1 platform is equipped with powerful CUDA-enabled GPUs which enables fast and parallel matrix multiplication. For future research work, mobile visual recognition systems may assume a 3-tier (mobile-edge-cloud) processing architecture. For example, the mobile device generates the query image, the edge unit adds privacy protection to the image, and the server processes the recognition.

Bibliography

[1] Perret, E. Heres how many digital photos will be taken in 2017. http://mylio.com/true-stories/next/announcing-2017-world-digital-photo-forecast. Accessed: 2016-09-23.

[2] Kiapour, M. H., Han, X., Lazebnik, S., Berg, A. C. & Berg, T. L. Where to buy it: Matching street clothing photos in online shops. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 3343–3351 (2015).

[3] Liu, Z., Yan, S., Luo, P., Wang, X. & Tang, X. Fashion landmark detection in the wild. *CoRR* **abs/1608.03049** (2016). URL http://arxiv.org/abs/1608.03049.

[4] http://www.flowerchecker.com/.

[5] Ruf, B., Kokiopoulou, E. & Detyniecki, M. Mobile museum guide based on fast sift recognition. In Detyniecki, M., Leiner, U. & Nrnberger, A. (eds.) *Adaptive Multimedia Retrieval. Identifying, Summarizing, and Recommending Image and Music*, vol. 5811 of *Lecture Notes in Computer Science*, 170–183 (Springer Berlin Heidelberg, 2010). URL http://dx.doi.org/10.1007/978-3-642-14758-6_14.

[6] Li, D. & Chuah, M. C. Emovis: An efficient mobile visual search system for landmark recognition. *2013 IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks* **00**, 53–60 (2013).

[7] Huang, Z., Hui, P., Peylo, C. & Chatzopoulos, D. Mobile augmented reality survey: a bottom-up approach. *CoRR* **abs/1309.4413** (2013). URL `http://arxiv.org/abs/1309.4413`.

[8] http://www.pokemongo.com/.

[9] https://www.fieldtripper.com/.

[10] Hani Altwaijry, M. M. & Belongie, S. Recognizing locations with google glass: A case study. *2014 IEEE Winter Conference on Applications of Computer Vision (WACV)* **00**, 167–174 (2014).

[11] Mandal, B. *et al. A Wearable Face Recognition System on Google Glass for Assisting Social Interactions*, 419–433 (Springer International Publishing, Cham, 2015). URL `http://dx.doi.org/10.1007/978-3-319-16634-6_31`.

[12] Henderson, S. & Feiner, S. Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE Transactions on Visualization and Computer Graphics* **17**, 1355–1368 (2011). URL `http://dx.doi.org/10.1109/TVCG.2010.245`.

[13] Chang, K.-E. *et al.* Development and behavioral pattern analysis of a mobile guide system with augmented reality for painting appreciation instruction in an art museum. *Computers and Education* **71**, 185 – 197 (2014). URL `http://www.sciencedirect.com/science/article/pii/S0360131513002868`.

[14] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A. & Oliva, A. Learning deep features for scene recognition using places database. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D. & Weinberger, K. Q. (eds.) *Advances in Neural Information Processing Systems 27*, 487–495 (Curran Associates, Inc., 2014). URL http://papers.nips.cc/paper/5349-learning-deep-features-for-scene-recognition-using-places-database.pdf.

[15] http://questvisual.com/.

[16] http://www.google.com/mobile/goggles/.

[17] http://camfindapp.com/.

[18] Huang, J. *et al.* A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, 225–238 (ACM, New York, NY, USA, 2012). URL http://doi.acm.org/10.1145/2307636.2307658.

[19] Sivic, J. & Zisserman, A. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, 1470– (IEEE Computer Society, Washington, DC, USA, 2003). URL http://dl.acm.org/citation.cfm?id=946247.946751.

[20] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* **60**, 91–110 (2004). URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[21] Bay, H., Ess, A., Tuytelaars, T. & Van Gool, L. Speeded-up robust features (surf). *Comput. Vis. Image Underst.* **110**, 346–359 (2008). URL http://dx.doi.org/10.1016/j.cviu.2007.09.014.

[22] Hartl, A., Schmalstieg, D. & Reitmayr, G. Client-side mobile visual search. In *VISAPP 2014 - Proceedings of the 9th International Conference on Computer Vision Theory and Applications* (2014).

[23] Thomee, B., Bakker, E. M. & Lew, M. S. Top-surf: A visual words toolkit. In *Proceedings of the International Conference on Multimedia*, MM '10, 1473–1476 (ACM, New York, NY, USA, 2010). URL http://doi.acm.org/10.1145/1873951.1874250.

[24] Chen, D. *et al.* Residual enhanced visual vector as a compact signature for mobile visual search. *Signal Processing* **93**, 2316 – 2327 (2013). URL http://www.sciencedirect.com/science/article/pii/S016516841200196X. Indexing of Large-Scale Multimedia Signals.

[25] Philbin, J., Chum, O., Isard, M., Sivic, J. & Zisserman, A. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8 (2007).

[26] Fischler, M. A. & Bolles, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**, 381–395 (1981). URL http://doi.acm.org/10.1145/358669.358692.

[27] Pedronette, D. C. G., Penatti, O. A. B., Calumby, R. T. & da S. Torres, R. Unsupervised distance learning by reciprocal knn distance for image retrieval. In *Proceedings of International Conference on Multimedia Retrieval*, ICMR '14, 345:345–345:352 (ACM, New York, NY, USA, 2014). URL http://doi.acm.org/10.1145/2578726.2578770.

[28] Chen, Y., Li, X., Dick, A. & Hill, R. Ranking consistency for image matching and object retrieval. *Pattern Recogn.* **47**, 1349–1360 (2014). URL http://dx.doi.org/10.1016/j.patcog.2013.09.011.

[29] Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25* (Curran Associates, Inc., 2012). URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[30] Szegedy, C. *et al.* Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)* (2015). URL http://arxiv.org/abs/1409.4842.

[31] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *CoRR* **abs/1512.03385** (2015). URL http://arxiv.org/abs/1512.03385.

[32] Girshick, R. B. Fast R-CNN. *CoRR* **abs/1504.08083** (2015). URL http://arxiv.org/abs/1504.08083.

[33] Ren, S., He, K., Girshick, R. B. & Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR* **abs/1506.01497** (2015). URL `http://arxiv.org/abs/1506.01497`.

[34] Redmon, J., Divvala, S. K., Girshick, R. B. & Farhadi, A. You only look once: Unified, real-time object detection. *CoRR* **abs/1506.02640** (2015). URL `http://arxiv.org/abs/1506.02640`.

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

[36] Kim, Y. *et al.* Compression of deep convolutional neural networks for fast and low power mobile applications. *CoRR* **abs/1511.06530** (2015). URL `http://arxiv.org/abs/1511.06530`.

[37] Denton, E. L., Zaremba, W., Bruna, J., Lecun, Y. & Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. & Weinberger, K. (eds.) *Advances in Neural Information Processing Systems 27*, 1269–1277 (Curran Associates, Inc., 2014).

[38] Hoffman, J. *et al.* One-shot adaptation of supervised deep convolutional models. *CoRR* **abs/1312.6204** (2013). URL `http://arxiv.org/abs/1312.6204`.

[39] Donahue, J. *et al.* Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR* **abs/1310.1531** (2013). URL `http://arxiv.org/abs/1310.1531`.

[40] Reyes, A. K., Caicedo, J. C. & Camargo, J. E. Fine-tuning deep convolutional networks for plant recognition. In *CLEF* (2015).

[41] Li, D. & Chuah, M. C. Emod: An efficient on-device mobile visual search system. In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, 25–36 (ACM, New York, NY, USA, 2015). URL http://doi.acm.org/10.1145/2713168.2713172.

[42] D. Li, N. V. D., T. Salonidis & Chuah, M. Deepcham: Collaborative edge-mediated adaptive deep learning for mobile object recognition. In *The First IEEE/ACM Symposium on Edge Computing*, SEC '16 (2016).

[43] Chen, X. & Koskela, M. Mobile visual search from dynamic image databases. In Heyden, A. & Kahl, F. (eds.) *Image Analysis*, vol. 6688 of *Lecture Notes in Computer Science*, 196–205 (Springer Berlin Heidelberg, 2011). URL http://dx.doi.org/10.1007/978-3-642-21227-7_19.

[44] Li, D. & Chuah, M. C. Emovis: An efficient mobile visual search system for landmark recognition. In *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth International Conference on*, 53–60 (2013).

[45] Kumhyr, D. Method for suspect identification using scanning of surveillance media. In *US Patent Application 10/185685* (2014). URL http://www.google.com/patents/US20040001142.

[46] Song, Y., Cai, W. & Deng, D. Hierarchical spatial matching for medical image retrieval. In *Proceedings of 2011 International ACM Workshop on Medical Multimedia Analysis and Retrieval* (2011).

[47] Yap, K.-H., Chen, T., Li, Z. & Wu, K. A comparative study of mobile-based landmark recognition techniques. *Intelligent Systems, IEEE* **25**, 48–57 (2010).

[48] Chen, D. & Girod, B. Memory-efficient image databases for mobile visual search. *MultiMedia, IEEE* **21**, 14–23 (2014).

[49] Chen, D. *et al.* Residual enhanced visual vector as a compact signature for mobile visual search. *Signal Process.* **93**, 2316–2327 (2013). URL `http://dx.doi.org/10.1016/j.sigpro.2012.06.005`.

[50] Sivic, J. & Zisserman, A. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, 1470– (IEEE Computer Society, Washington, DC, USA, 2003). URL `http://dl.acm.org/citation.cfm?id=946247.946751`.

[51] Nister, D. & Stewenius, H. Scalable recognition with a vocabulary tree. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, 2161–2168 (IEEE Computer Society, Washington, DC, USA, 2006). URL `http://dx.doi.org/10.1109/CVPR.2006.264`.

[52] Philbin, J., Chum, O., Isard, M., Sivic, J. & Zisserman, A. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 1–8 (2007).

[53] Turcot, P. & Lowe, D. Better matching with fewer features: The selection of useful features in large database recognition problems. In *Computer Vision*

Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, 2109–2116 (2009).

[54] Knopp, J., Sivic, J. & Pajdla, T. Avoiding confusing features in place recognition. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, 748–761 (Springer-Verlag, Berlin, Heidelberg, 2010). URL http://dl.acm.org/citation.cfm?id=1886063.1886120.

[55] Philbin, J., Isard, M., Sivic, J. & Zisserman, A. Descriptor learning for efficient retrieval. In Daniilidis, K., Maragos, P. & Paragios, N. (eds.) *Computer Vision ECCV 2010*, vol. 6313 of *Lecture Notes in Computer Science*, 677–691 (Springer Berlin Heidelberg, 2010). URL http://dx.doi.org/10.1007/978-3-642-15558-1_49.

[56] Fischler, M. A. & Bolles, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**, 381–395 (1981). URL http://doi.acm.org/10.1145/358669.358692.

[57] Chen, Y., Li, X., Dick, A. & Hill, R. Ranking consistency for image matching and object retrieval. *Pattern Recognition* **47**, 1349 – 1360 (2014). URL http://www.sciencedirect.com/science/article/pii/S003132031300383X. Handwriting Recognition and other {PR} Applications.

[58] Chum, O., Philbin, J. & Zisserman, A. Near duplicate image detection: min-hash and tf-idf weighting. In *British Machine Vision Conference* (2008).

[59] Sanchez, J. & Perronnin, F. High-dimensional signature compression for

large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 1665–1672 (2011).

[60] Jegou, H., Douze, M. & Schmid, C. Packing bag-of-features. In *Computer Vision, 2009 IEEE 12th International Conference on*, 2357–2364 (2009).

[61] Perd'och, M., Chum, O. & Matas, J. Efficient representation of local geometry for large scale object retrieval. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 9–16 (2009).

[62] Mikolajczyk, K. *et al.* A comparison of affine region detectors. *Int. J. Comput. Vision* **65**, 43–72 (2005). URL `http://dx.doi.org/10.1007/s11263-005-3848-x`.

[63] Muja, M. & Lowe, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, 331–340 (INSTICC Press, 2009).

[64] Silpa-Anan, C. & Hartley, R. Optimised kd-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 1–8 (2008).

[65] Beis, J. & Lowe, D. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, 1000–1006 (1997).

[66] Keogh, E., Lin, J. & Fu, A. Hot sax: Efficiently finding the most unusual time series subsequence. In *Proceedings of the Fifth IEEE International Conference*

on Data Mining, ICDM '05, 226–233 (IEEE Computer Society, Washington, DC, USA, 2005). URL http://dx.doi.org/10.1109/ICDM.2005.79.

[67] Qin, D., Gammeter, S., Bossard, L., Quack, T. & Van Gool, L. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 777–784 (2011).

[68] Pedronette, D. C. G., Penatti, O. A. B., Calumby, R. T. & da S. Torres, R. Unsupervised distance learning by reciprocal knn distance for image retrieval. In *Proceedings of International Conference on Multimedia Retrieval*, ICMR '14, 345:345–345:352 (ACM, New York, NY, USA, 2014). URL http://doi.acm.org/10.1145/2578726.2578770.

[69] http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html/.

[70] http://www.robots.ox.ac.uk/ vgg/data/oxbuildings/.

[71] http://www.robots.ox.ac.uk/ vgg/data/parisbuildings/.

[72] http://mickey.cse.lehigh.edu/lubud/.

[73] Turpin, A. & Scholer, F. User performance versus precision measures for simple search tasks. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, 11–18 (ACM, New York, NY, USA, 2006). URL http://doi.acm.org/10.1145/1148170.1148176.

[74] Manjunath, B., Ohm, J.-R., Vasudevan, V. & Yamada, A. Color and texture descriptors. *Circuits and Systems for Video Technology, IEEE Transactions on* **11**, 703–715 (2001).

[75] Chatzichristofis, S. A. & Boutalis, Y. S. Cedd: color and edge directivity descriptor: a compact descriptor for image indexing and retrieval. In *Proceedings of the 6th international conference on Computer vision systems*, ICVS'08, 312–322 (2008).

[76] Chun, Y. D., Kim, N. C. & Jang, I. H. Content-based image retrieval using multiresolution color and texture features. *Multimedia, IEEE Transactions on* **10**, 1073–1084 (2008).

[77] Yang, Y. & Newsam, S. Geographic image retrieval using local invariant features. *Geoscience and Remote Sensing, IEEE Transactions on* **51**, 818–832 (2013).

[78] https://github.com/EsotericSoftware/kryo/.

[79] Jegou, H., Harzallah, H. & Schmid, C. A contextual dissimilarity measure for accurate and efficient image search. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 1–8 (2007).

[80] Jegou, H., Douze, M. & Schmid, C. Hamming embedding and weak geometric consistency for large scale image search. In Forsyth, D., Torr, P. & Zisserman, A. (eds.) *Computer Vision ECCV 2008*, 304–317 (Springer Berlin Heidelberg, 2008). URL `http://dx.doi.org/10.1007/978-3-540-88682-2_24`.

[81] Jegou, H., Douze, M. & Schmid, C. On the burstiness of visual elements. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 1169–1176 (2009).

[82] Shen, X., Lin, Z., Brandt, J., Avidan, S. & Wu, Y. Object retrieval and localization with spatially-constrained similarity measure and k-nn re-ranking. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 3013–3020 (2012).

[83] Lane, N. D. *et al.* Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 1–12 (2016).

[84] Radu, V. *et al.* Towards Multimodal Deep Learning for Activity Recognition on Mobile Devices. In *Proc. UbiComp'16* (Heidelberg, Germany, 2016).

[85] Fu, J., Mei, T., Yang, K., Lu, H. & Rui, Y. Tagging Personal Photos with Transfer Deep Learning. In *Proc. WWW'15* (Florence, Italy, 2015).

[86] Liu, C. *et al.* Lasagna: Towards Deep Hierarchical Understanding and Searching over Mobile Sensing Data. In *Proc. MobiCom'16* (New York City, NY, 2016).

[87] Lane, N. D. & Georgiev, P. Can Deep Learning Revolutionize Mobile Sensing? In *Proc. HotMobile'15* (Santa Fe, NM, 2015).

[88] Han, S. *et al.* MCDNN: An Approximation-Based Execution Framework for Deep Stream Processing Under Resource Constrains. In *Proc. MobiSys'16* (Singapore, 2016).

[89] Chen, T. Y.-H., Ravindranath, L., Deng, S., Bahl, P. & Balakrishnan, H. Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices. In *Proc. SenSys'15* (Seoul, South Korea, 2015).

[90] Mastercard Introduces Facial Recognition Mobile Payments in Europe. `www.biometricupdate.com/201610/mastercard-introduces-facial-recognition-mobile-payments-in-europe`.

[91] Sinha, A., Choi, C. & Ramani, K. DeepHand: Robust Hand Pose Estimation by Completing a Matrix Imputed with Deep Features. In *Proc. CVPR'16* (Las Vegas, 2016).

[92] Augmented Reality. `https://en.wikipedia.org/wiki/Augmented_reality`.

[93] Rastegari, M., Ordonez, V., Redmon, J. & Farhadi, A. XNOR-NET: ImageNet Classification Using Binary Convolutional Neural Networks. *https://arxiv.org/pdf/1603.05279.pdf* (2016).

[94] Huang, J., Badam, A., Chandra, R. & Nightingale, E. B. WearDrive: Fast and Energy-Efficient Storage for Wearables. In *Proc. USENIX ATC'15* (Santa Clara, CA, 2015).

[95] Varadarajan, V., Zhang, Y., Ristenpart, T. & Swift, M. A Placement Vulnerability Study in Multi-Tenant Public Clouds. In *Proc. USENIX Security'15* (Washington, D.C., 2015).

[96] Baumann, A., Peinado, M. & Hunt, G. Shielding Applications from An Untrusted Cloud with Haven. In *Proc. OSDI'14* (Broomfield, CO, 2014).

[97] Cost to Build a Mobile App: A Survey. https://clutch.co/app-developers/resources/cost-build-mobile-app-survey.

[98] Cloud Encryption: Using Data Encryption in the Cloud. http://www.tomsitpro.com/articles/cloud-data-encryption,2-913.html.

[99] Cao, Y. & Yang, J. Towards Making Systems Forget with Machine Unlearning. In *Proc. Oakland'15* (San Jose, CA, 2015).

[100] Shokri, R. & Shmatikov, V. Privacy-Preserving Deep Learning. In *Proc. CCS'15* (Denver, Colorado, 2015).

[101] Hunt, T., Zhu, Z., Xu, Y., Peter, S. & Witchel, E. Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data. In *Proc. OSDI'16* (Savannah, GA, 2016).

[102] Battery Density Trends Research, Argonne National Laboratory. http://ec.europa.eu/dgs/jrc/downloads/events/20130926-eco-industries/20130926-eco-industries-miller.pdf.

[103] LiKamWa, R., Hou, Y., Gao, J., Polansky, M. & Zhong, L. RedEye: Analog ConvNet Image Sensor Architecture for Continuous Mobile Vision. In *Proc. ISCA'16* (Seoul, Korea, 2016).

[104] Dunn, A. M. *et al.* Eternal Sunshine of the Splotless Machine: Protecting Privacy with Ephemeral Channels. In *Proc. OSDI'12* (Hollywood, CA, 2012).

[105] Roy, I., Setty, S. T., Kilzer, A., Shmatikov, V. & Witchel, E. Airavat: Security and Privacy for MapReduce. In *Proc. NSDI'10* (San Jose, CA, 2010).

[106] Yuan, J., Yu, S. & Guo, L. Seisa: Secure and efficient encrypted image search with access control. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2083–2091 (2015).

[107] Sun, W. *et al.* Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. *IEEE Transactions on Parallel and Distributed Systems* **25**, 3025–3035 (2014).

[108] Cao, N., Wang, C., Li, M., Ren, K. & Lou, W. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems* **25**, 222–233 (2014).

[109] Le Cun, B. B. *et al.* Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems* (1990).

[110] Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105 (2012).

[111] Farabet, C., Couprie, C., Najman, L. & LeCun, Y. Learning hierarchical features for scene labeling. *PAMI* **35**, 1915–1929 (2013).

[112] Ciresan, D., Giusti, A., Gambardella, L. M. & Schmidhuber, J. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, 2843–2851 (2012).

[113] Zeiler, M. D. & Fergus, R. Visualizing and understanding convolutional neural networks. *arXiv preprint arXiv:1311.2901* (2013).

[114] Donahue, J. *et al.* Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR* **abs/1310.1531** (2013). URL `http://arxiv.org/abs/1310.1531`.

[115] Han, S., Mao, H. & Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR* **abs/1510.00149** (2015). URL `http://arxiv.org/abs/1510.00149`.

[116] Huffman, D. A. A method for the construction of minimum-redundancy codes. *Resonance* **11**, 91–99 (2006). URL `http://dx.doi.org/10.1007/BF02837279`.

[117] Amazon Machine Learning – Android Developer Guide. `http://docs.aws.amazon.com/mobile/sdkforandroid/developerguide/getting-started-machine-learning.html`.

[118] Monsoon Power Monitor. `http://www.msoon.com/LabEquipment/PowerMonitor/`.

[119] Kolda, T. G. & Bader, B. W. Tensor decompositions and applications. *SIAM Review* **51**, 455–500 (2009). URL `http://dx.doi.org/10.1137/07070111X`. `http://dx.doi.org/10.1137/07070111X`.

[120] Deng, J. *et al.* ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09* (2009).

[121] Chon, Y., Lane, N. D., Li, F., Cha, H. & Zhao, F. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, 481–490 (ACM, New York, NY, USA, 2012). URL http://doi.acm.org/10.1145/2370216.2370288.

[122] Parkhi, O. M., Vedaldi, A. & Zisserman, A. Deep face recognition. In *British Machine Vision Conference* (2015).

[123] Ng, H. W. & Winkler, S. A data-driven approach to cleaning large face datasets. In *2014 IEEE International Conference on Image Processing (ICIP)*, 343–347 (2014).

[124] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[125] Szegedy, C. *et al.* Going Deeper with Convolutions. In *Proc. CVPR'15* (Boston, MA, 2015).

[126] Simonyan, K. & Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proc. ICLR'15* (San Diego, CA, 2015).

[127] Goodfellow, I. *et al.* Generative adversarial nets. In Ghahramani, Z.,

Welling, M., Cortes, C., Lawrence, N. D. & Weinberger, K. Q. (eds.) *Advances in Neural Information Processing Systems 27*, 2672–2680 (Curran Associates, Inc., 2014). URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[128] https://github.com/sh1r0/caffe-android-lib.

[129] Abadi, M. & et al. TensorFlow: Large-scale machine learning on heterogeneous systems (2015). URL http://tensorflow.org/. Software available from tensorflow.org.

[130] https://github.com/jetpacapp/deepbeliefsdk.

[131] Baidu is Bringing Intelligent AR to the Masses. https://www.technologyreview.com/s/602091/baidu-is-bringing-intelligent-ar-to-the-masses/.

[132] Han, S., Mao, H. & Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR* **abs/1510.00149** (2015). URL http://arxiv.org/abs/1510.00149.

[133] Kim, Y. *et al.* Compression of deep convolutional neural networks for fast and low power mobile applications. *CoRR* **abs/1511.06530** (2015). URL http://arxiv.org/abs/1511.06530.

[134] Iandola, F. N. *et al.* Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR* **abs/1602.07360** (2016). URL http://arxiv.org/abs/1602.07360.

[135] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. Rethinking the inception architecture for computer vision. *CoRR* **abs/1512.00567** (2015). URL `http://arxiv.org/abs/1512.00567`.

[136] Chen, T. *et al.* DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning. In *Proc. ASPLOS'14* (Salt Lake City, Utah, 2014).

[137] Liu, S. *et al.* Cambricon: An Instruction Set Architecuture for Neural Networks. In *Proc. ISCA'16* (Seoul, Korea, 2016).

[138] Kumar, K., Liu, J., Lu, Y.-H. & Bhargava, B. A Survey of Computation Offloading for Mobile Systems. *Mobile Network Application* (2012).

[139] Chun, B.-G. & Maniatis, P. Augmented Smartphone Applications Through Clone Cloud Execution. In *Proc. HotOS'09* (Monte Verita, Switzerland, 2009).

[140] Cuervo, E. *et al.* MAUI: Making Smartphones Last Longer with Code Offload. In *Proc. MobiSys'10* (San Francisco, CA, 2010).

[141] Ra, M.-R. *et al.* Odessa: Enabling Interactive Perception Applications on Mobile Devices. In *Proc. MobiSys'11* (Bethesda, Maryland, 2011).

[142] Ra, M., Govindan, R. & Ortega, A. P3: Toward privacy-preserving photo sharing. In *Proc. USENIX NSDI* (2013).

[143] Unterweger, A. & Uhl, A. Length-preserving bit-stream-based jpeg encryption. In *Proc. ACM MMSec* (2012).

[144] Wu, C. & Kuo, C. Design of integrated multimedia compression and encryption systems. *Multimedia, IEEE Trans. on* **7**, 828–839 (2005).

[145] Narayan, A., Feldman, A., Papadimitriou, A. & Haeberlen, A. Verifiable differential privacy. In *EuroSys 2015* (2015).

[146] Jain, P., Manweiler, J. & Chowdhury, R. Overlay: Practical Mobile Augmented Reality. In *Proc. ACM MobiSys* (Florence, Italy, 2015).

[147] Takacs, G. *et al.* Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval*, MIR '08, 427–434 (ACM, New York, NY, USA, 2008). URL `http://doi.acm.org/10.1145/1460096.1460165`.

[148] Li, D., Chuah, M.-C. & Tian, L. Demo: Lehigh explorer augmented campus tour (lact). In *Proceedings of the 2014 Workshop on Mobile Augmented Reality and Robotic Technology-based Systems*, MARS '14, 15–16 (ACM, New York, NY, USA, 2014). URL `http://doi.acm.org/10.1145/2609829.2609831`.

[149] Kahn, P. H., Jr., Friedman, B., Perez-Granados, D. R. & Freier, N. G. Robotic pets in the lives of preschool children. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '04, 1449–1452 (ACM, New York, NY, USA, 2004). URL `http://doi.acm.org/10.1145/985921.986087`.

[150] Jia, Y. *et al.* Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093* (2014).

[151] Satyanarayanan, M., Bahl, P., Caceres, R. & Davies, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing* **8**, 14–23 (2009).

[152] Satyanarayanan, M. *et al.* Cloudlets: at the leading edge of mobile-cloud convergence. In *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*, 1–9 (2014).

[153] Verbelen, T., Simoens, P., De Turck, F. & Dhoedt, B. Cloudlets: Bringing the cloud to the mobile user. In *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, MCS '12, 29–36 (ACM, New York, NY, USA, 2012). URL `http://doi.acm.org/10.1145/2307849.2307858`.

[154] Bonomi, F., Milito, R., Zhu, J. & Addepalli, S. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, 13–16 (ACM, New York, NY, USA, 2012). URL `http://doi.acm.org/10.1145/2342509.2342513`.

[155] Castro, D. *et al.* Predicting daily activities from egocentric images using deep learning. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, ISWC '15, 75–82 (ACM, New York, NY, USA, 2015). URL `http://doi.acm.org/10.1145/2802083.2808398`.

[156] Furnari, A., Farinella, G. M. & Battiato, S. Recognizing personal contexts from egocentric images. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, 393–401 (2015).

[157] Patel, V. M., Gopalan, R., Li, R. & Chellappa, R. Visual domain adaptation:

A survey of recent advances. *Signal Processing Magazine, IEEE* **32**, 53–69 (2015).

[158] Lane, N. D. & Georgiev, P. Can deep learning revolutionize mobile sensing? In *Proc. ACM HotMobile*, 117–122 (ACM, 2015).

[159] Lane, N. D., Georgiev, P. & Qendro, L. Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proc. ACM UBICOMP*, 283–294 (ACM, 2015).

[160] Brodley, C. E. & Friedl, M. A. Identifying mislabeled training data. *CoRR* **abs/1106.0219** (2011). URL `http://arxiv.org/abs/1106.0219`.

[161] Mensink, T., Verbeek, J. & Csurka, G. Learning structured prediction models for interactive image labeling. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 833–840 (IEEE, 2011).

[162] Mensink, T., Verbeek, J. & Csurka, G. Tree-structured crf models for interactive image labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **35**, 476–489 (2013).

[163] Wenyin, L. *et al.* Semi-automatic image annotation. In *Proc. of Human-Computer Interaction-Interact*, 326–333 (2001).

[164] Zhang, L., Tong, Y. & Ji, Q. Active image labeling and its application to facial action labeling. In *Computer Vision–ECCV 2008*, 706–719 (Springer, 2008).

[165] Zitnick, C. L. & Dollár, P. Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014*, 391–405 (Springer, 2014).

[166] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T. & Smeulders, A. W. M. Selective search for object recognition. *International Journal of Computer Vision* **104**, 154–171 (2013). URL `https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013`.

[167] Girshick, R. B., Donahue, J., Darrell, T. & Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR* **abs/1311.2524** (2013). URL `http://arxiv.org/abs/1311.2524`.

[168] Yi, S., Li, C. & Li, Q. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, Mobidata '15, 37–42 (ACM, New York, NY, USA, 2015). URL `http://doi.acm.org/10.1145/2757384.2757397`.

[169] Chatzichristofis, S. & Boutalis, Y. Cedd: Color and edge directivity descriptor: A compact descriptor for image indexing and retrieval. In *Computer Vision Systems*, vol. 5008, 312–322 (Springer Berlin Heidelberg, 2008). URL `http://dx.doi.org/10.1007/978-3-540-79547-6_30`.

[170] Russakovsky, O. *et al.* ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* **115**, 211–252 (2015).

[171] Wan, J. *et al.* Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22Nd ACM International Conference*

on Multimedia, MM '14, 157–166 (ACM, New York, NY, USA, 2014). URL http://doi.acm.org/10.1145/2647868.2654948.

[172] Ortega, R., Avery, J. & Frederick, R. Search query autocompletion (2003). URL https://www.google.si/patents/US6564213. US Patent 6,564,213.

[173] Lodi, S., anculef, R. & Sartori, C. *Single-Pass Distributed Learning of Multi-Class SVMs using Core-Sets*, 257–268 (SIAM, 2010). URL http://epubs.siam.org/doi/abs/10.1137/1.9781611972801.23. http://epubs.siam.org/doi/pdf/10.1137/1.9781611972801.23.

[174] https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet/.

[175] https://github.com/pdollar/edges.

[176] Fan, R.-E., Chen, P.-H. & Lin, C.-J. Working set selection using second order information for training support vector machines. *J. Mach. Learn. Res.* **6**, 1889–1918 (2005). URL http://dl.acm.org/citation.cfm?id=1046920.1194907.

[177] Arthur, D. & Vassilvitskii, S. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, 1027–1035 (Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007). URL http://dl.acm.org/citation.cfm?id=1283383.1283494.

[178] Lin, M., Chen, Q. & Yan, S. Network in network. *CoRR* **abs/1312.4400** (2013). URL http://arxiv.org/abs/1312.4400.

[179] Zhang, L. *et al.* Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES/ISSS '10, 105–114 (ACM, New York, NY, USA, 2010). URL http://doi.acm.org/10.1145/1878961.1878982.

[180] Cauwenberghs, G. & Poggio, T. Incremental and decremental support vector machine learning. In *Adv. Neural Information Processing Systems*, NIPS '2000 (MIT Press, 2000).

[181] Raykar, V. C. & Yu, S. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *J. Mach. Learn. Res.* **13**, 491–518 (2012). URL http://dl.acm.org/citation.cfm?id=2503308.2188401.

[182] Bardenet, R., Brendel, M., Kgl, B. & Sebag, M. Collaborative hyperparameter tuning. In Dasgupta, S. & Mcallester, D. (eds.) *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, vol. 28, 199–207 (JMLR Workshop and Conference Proceedings, 2013). URL http://jmlr.csail.mit.edu/proceedings/papers/v28/bardenet13.pdf.

[183] Morvant, E., Habrard, A. & Ayache, S. *Majority Vote of Diverse Classifiers for Late Fusion*, 153–162 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2014). URL http://dx.doi.org/10.1007/978-3-662-44415-3_16.

Vita

1987	Born in Tengzhou, Shandong Province, China.
2005	Graduated from Zaozhuang No. 3 High School, Zaozhuang, China.
2009	B.S. in Software Engineering, Beihang University, China.
2011	M.S. in Computer Science, The University of Alabama.
2012-2017	Graduate study in Department of Computer Science and Engineering, Lehigh University.
2015	Research internship at IBM Research.
2016	Research internship at Samsung Research America.
2017	Joined Samsung Research America as Senior Research Engineer.

2011	Dawei Li, Xiaoyan Hong, Jason Bowman. **Evaluation of Security Vulnerabilities by Using ProtoGENI as a Launchpad.** IEEE Globecom.
2013	Dawei Li, Xiaoyan Hong, Darwin Witt. **ProtoGENI, A Prototype GENI Under Security Vulnerabilities : An Experiment-Based Security Study.** IEEE Systems Journal.
2013	Dawei Li, Mooi Choo Chuah. **SCOM - A Scalable Content Centric Network Architecture with Mobility Support.** IEEE Mobile Ad hoc Sensor Networks (MSN).

2013 Dawei Li, Mooi Choo Chuah. **EMOVIS - An Efficient Mobile Visual Search System for Landmark Recognition.** IEEE Mobile Ad hoc Sensor Networks (MSN).

2013 Ziyuan Qin, Dawei Li, Mooi Choo Chuah. **Lehigh Explorer: A Real Time Video Streaming Application with Mobility Support for Content Centric Networks.** IEEE Mobile Ad hoc Sensor Networks (MSN).

2014 Dawei Li, Mooi Choo Chuah, Ziyuan Qin. **Design and Evaluation of Real-time Video Streaming Service with Mobitlity Support for Content Centric Networks.** Ad Hoc & Sencor Wireless Networks (AHSWN) Journal.

2014 Dawei Li, Mooi Choo Chuah, Li Tian. **Demo: Lehigh Explorer Augmented Campus Tour (LACT).** ACM MobiSys Workshop on Mobile Augmented Reality and Robotics-based Technology System.

2015 Dawei Li, Mooi Choo Chuah. **EMOD: an efficient on-device mobile visual search system.** ACM Multimedia Systems Conference (MMSys).

2015 Dawei Li, Mooi Choo Chuah. **A Novel Unsupervised 2-Stage k-NN Re-Ranking Algorithm for Image Retrieval.** IEEE International Symposium on Multimedia (ISM).

2016 Dawei Li, Mooi Choo Chuah. **Accurate Image Retrieval with Unsupervised 2-Stage k-NN Re-Ranking.** International Journal of Multimedia Data Engineering and Management (IJMDEM).

2016 Dawei Li, Theodoros Salonidis, Nirmit Desai, Mooi Choo Chuah. **DeepCham: Collaborative Edge-Mediated Adaptive Deep Learning for Mobile Object Recognition.** IEEE/ACM Symposium on Edge Computing.