CrowdVision: A Computing Platform for Video Crowdprocessing Using Deep Learning

Seminar Report

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

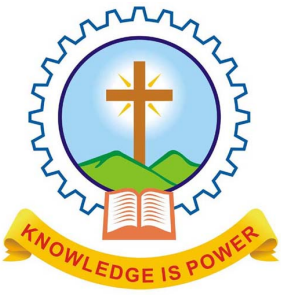
COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Submitted By

RAHUL A



Department of Computer Science & Engineering
Mar Athanasius College Of Engineering
Kothamangalam

CrowdVision: A Computing Platform for Video Crowdprocessing Using Deep Learning

Seminar Report

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

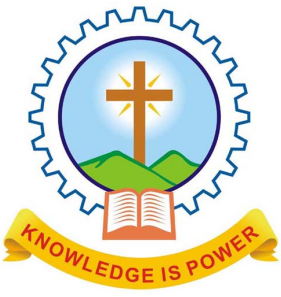
COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Submitted By

RAHUL A



Department of Computer Science & Engineering
Mar Athanasius College Of Engineering
Kothamangalam

Mar Athanasius College of Engineering

KOTHAMANGALAM

CrowdVision: A Computing Platform for Video Crowdprocessing Using Deep Learning

Bonafide record of seminar done by

RAHUL A
Register Number: LMAC15CS065

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

.....
Prof. Joby George
Faculty Guide

.....
Dr. Surekha Mariam Varghese
HOD

.....
Prof. Neethu Subash
Faculty Guide

ACKNOWLEDGEMENT

First and foremost, I sincerely thank the 'God Almighty' for his grace for the successful and timely completion of the seminar.

I express my sincere gratitude and thanks to Dr. Solly George T, Principal and Dr.Surekha Mariam Varghese, Head of the Department for providing the necessary facilities and their encouragement and support

I owe special thanks to the staff in charge Mr.Joby George, Mrs.Neethu Subash and Mrs. Joby for their corrections, suggestions and sincere efforts to coordinate the seminar under a right schedule

I express my sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in helping me to conduct this seminar.

Finally I would like to acknowledge the heart felt efforts, comments, criticisms, cooperation and tremendous support given to me by my dear friends during the preparation of the seminar and also during the presentation without whose support this would have been all the more difficult to accomplish.

ABSTRACT

Mobile devices such as smartphones are enabling users to generate and share videos. In some cases, these videos may contain valuable information, which can be exploited for a variety of purposes. However, instead of centrally collecting and processing videos for information retrieval, we consider crowd processing videos, where each mobile device locally processes stored videos. While the computational capability of mobile devices continues to improve, processing videos using deep learning, i.e., convolutional neural networks, is still a demanding task for mobile devices. To this end, we design and build Crowd Vision, a computing platform that enables mobile devices to crowd process videos using deep learning in a distributed and energy-efcient manner leveraging cloud ofoad. Crowd Vision can quickly and efciently process videos with ofoad under various settings and different network connections and greatly outperform the existing computation ofoad frame work . In doing so Crowd Vision tackles several challenges, how to exploit the characteristics of the computing of deep learning for video processing how to parallelize processing and ofoading for acceleration, how to optimize both time and energy at runtime by just determining the right moments to ofoad.

Contents

List of Figures

LIST OF ABBREVIATION

CNN	Convolutional neural networks
CUDA	Compute Unified Device Architecture
GPU	Graphical Proccessing

Chapter 1

Introduction

MOBILE devices such as smartphones are enabling users to generate and share videos with increasing rates. In some cases, these videos may contain valuable information, which can be exploited for a variety of purposes.

In this paper, we consider a video classication problem where a task issuer asks the crowd to identify relevant videos about a specic object or target. This requires object detection to be performed on videos to detect these objects of interest within the frames of the videos. The limitations of mobile devices for these types of applications are well known. The most obvious challenge is the computational requirement. Although it continues to improve, video processing using deep learning, i.e., Convolutional Neural Networks (CNNs), is still a demanding task for mobile devices .

We anticipate that video processing can be performed within a network of mobile devices and a powerful cloud environment. Individuals have the option to process the videos locally or ofoad them to a highly capable computing unit in the cloud. Coupled with the computation limitation is the cost to transmit, whether it be via WiFi or cellular (4G LTE). Transmitting videos to the cloud over wireless links consumes considerable energy. Additionally, when using cellular networks, there may be data budgets that limit data transfer without incurring extra expenses. As a result, we consider the coupled problem of constrained resources of computing, data transmission, and battery.

We anticipate that video processing can be performed within a network of mobile devices and a powerful cloud environment. Individuals have the option to process the videos locally or ofoad them to ahighly capable computing unit in the cloud. Coupled

with the computation limitation is the cost to transmit, whether it be via WiFi or cellular (4G LTE). Transmitting videos to the cloud over wireless links consumes considerable energy. Additionally, when using cellular networks, there may be data budgets that limit data transfer without incurring extra expenses. As a result, we consider the coupled problem of constrained resources of computing, data transmission, and battery.

Due to these limitations, users may be reluctant to participate in such requested video processing tasks. However, users could be motivated by various incentive mechanisms, such as , which are not the focus of this paper. For this paper, we consider a variation on a crowdsensing-type scenario, crowdprocessing . Instead of users collecting and sharing data to solve a larger problem, we explore the computing capability of mobile devices and allow individuals to process some (or all) of the data rather than having some centralized entity process everything .

We design and build CrowdVision, a computing platform that enables mobile devices to crowdprocess videos using deep learning in a distributed and energy-efcient manner leveraging cloud ofoad. In doing so we tackle several challenges. First, to design a computing platform specically for video processing using deep learning, we should take into account the characteristics of the computing of deep learning. By deploying and measuring the computing of CNNs on mobile devices, we identify batch processing, which makes deep learning different from common computational tasks and can be exploited for computing acceleration. Second, as frames extracted from a video arrive at a certain rate, to take advantage of the batch processing towards optimizing the processing time we need to determine when to perform each batch with how many frames. However, the problem turns out to be a NP-hard problem. By carefully balancing the waiting time for frames to be available and the processing time incurred by each additional processing batch, we are able to determine the batch processing to optimize the processing parallelized by ofoading, even with an energy constraint that is imposed by the user. Third, When the data rate for ofoading varies widely, it is hard for mobile devices to acknowledge when ofoad benets. By correlating signal strength, data rate, and power based on ofoading attempts, we are able to sense and seize the right mo-

ments when ofoading benets both processing time and energy at runtime. To the best of our knowledge, this is the rst work that takes into consideration batching processing in computational ofoad for the computing of deep learning.

We envision CrowdVision to be useful for a variety of applications that require content information of videos from the crowd. One common use case is emergency response situations, where municipal agencies ask the public to assist in identifying terrorists or criminals through scanning of their videos, as the FBI did after the Boston Marathon bombing. CrowdVision could handle this request in a smart and automatic way; i.e., it rst lters videos based on location and timestamp, and then collectively performs deep learning to nd the object of interest.

Chapter 2

RELATED WORK

T here are several examples of crowdsensing frameworks similar to CrowdVision. Medusa is a platform that performs crowdsensing tasks through the transfer and processing of media (text, images, videos). Pickle is a crowdsensing application to conduct collaborative learning. GigaSight is a cloud architecture for continuous collection of crowd-sourced video from mobile devices. In contrast to these frameworks,CrowdVision is a computing platform for video crowdprocessing using deep learning rather than the system or platform that motivates users and collects sensed data. For the crowdprocessing task, CrowdVision attempts to optimize the performance of the task execution while considering the energy usage and data usage of the participating mobile devices through computation ofoad.

There is a large body of work that studies computation ofoad for mobile devices. These can be summarily classied into two categories: building general models for computation ofoad such as and computation ofoad based applications such as MAUI investigates code ofoad to reserve energy for computation, while Hermes investigates optimal task assignment in mobile ofoading.The feasibility of computation ofoad is studied in term of communications networks.COSMOS is presented in,which bridges the gap between quick response requests by mobile devices and the long setup time of cloud frameworks. Resource allocation for mobile-edge computation ofoading in cellular networks is considered in Computation ofoad has been considered and implemented

in various applications, e.g., for interactive perception applications, social sensing and wearables. In addition to ofoading, Tango considers application replication on mobile devices and servers, where the active instance of the application switches between the two systems to accelerate network intensive applications.These existing works on computation ofoad consider regular computing tasks. However, as will be discussed in the next section, the computing of deep learning has the unique characteristic (i.e., batch processing) that differentiates it from regular computing tasks. Thus, these works cannot be applied effectively to our application. Moreover, unlike traditional ofoading problems where the cloud has to return the result of the computation to the mobile device, in crowd processing, mobile devices are queried from the cloud and the results are gathered at the cloud. This makes the problem different from these existing works.

Optimizing the computing of deep learning applications on mobile devices has been recently investigated by compressing parameters of CNNs , by distributing computation to heterogeneous processors on-board by trading off accuracy and resource usage, by mobile GPUs, and by ne-tuned cp-decomposition. However, currently, these approaches require custom software or hardware. Nevertheless, these potential processing approaches can be easily integrated with CrowdVision whenever they are available for off-the-shelf mobile devices.

Splitting the computing of CNN layers between mobile device and cloud is considered in However, based our experiments on AlexNet, VGGNet, GoogleNet and ResNet,the most intensive layer share the rst few layers of CNNs and later layers cost much less. Moreover, the separation of the layers will break the computing pipeline of CNNs, which could cost longer compute time, especially on GPUs.

Detecting objects in real-time for augmented reality applications is considered in, where the ofoad decision is made for each individual frame. However, we consider videos stored on mobile devices, which means ofoading decision is made on a large

number of frames and by determining the optimal batch processing. Therefore, the simple policy does not work in our case. In addition, energy-efcient scheduling for real-time systems using deep reinforcement learning is investigated.A more detailed review of deep learning on big data is given in

Chapter 3

OVERVIEW

CrowdVision is a distributed computing platform that enables mobile devices to participate in the crowd processing of videos. In this environment, a task issuer initiates a query by which mobile devices are requested to identify some object of interest within its locally stored videos. We consider a crowd processing approach to perform object detection/classication of videos. This task includes ltering of videos based on meta data, and then processing the videos to perform object detection (note that we currently do not consider multi-frame video classication. This takes several steps as described below. Caffe , a deep learning framework, is currently employed to perform object detection on frames extracted from videos. Although CrowdVision is currently built on top of Caffe,it is compatible with other deep learning frameworks, e.g., Torch and TensorFlow.

The details of the query can be of varying specicity (e.g., nd people wearing red hats walking a dog in the park on Tuesday evening). These details may allow for the individuals to lter out irrelevant videos (based on location or time stamp). At the expense of some energy usage, the user can ofoad the videos to the cloud, where high performance computing can quickly process the videos without energy usage concerns. Alternatively, the user can process some of the frames locally and ofoad others

Once these videos are processed using deep learning either locally on the mobile device or remotely on the cloud, the task issuer will receive information about the videos related to the query. For frames that are processed on the mobile devices, the user will

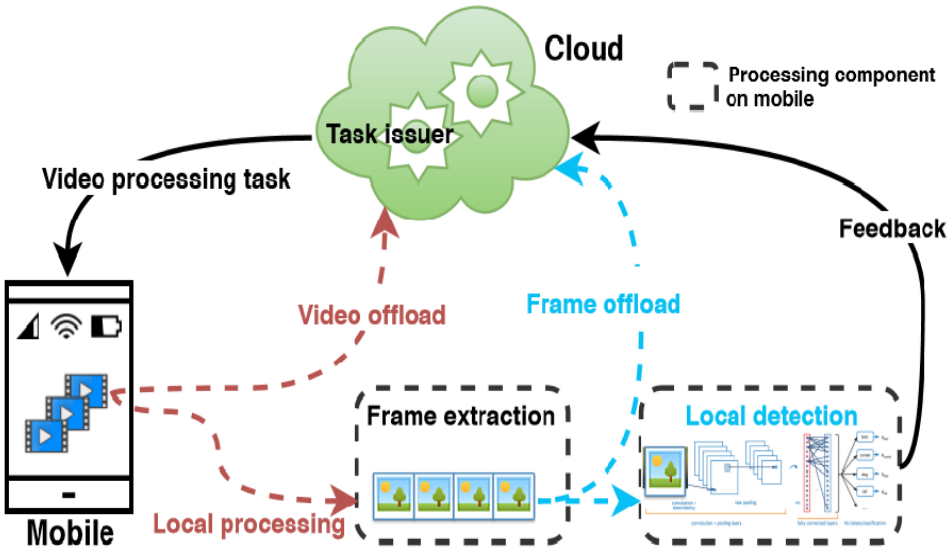


Fig. 3.1: Overview of CrowdVision

forward either the tags, the frames of interest, or the entire video to the cloud.

Two important aspects of this process are not in the scope of this paper. First, incentive mechanisms for crowd processing are important but not the focus of this paper. Second, participation in the task may reveal personal information and intrude on users privacy, but users are allowed to lter out their personal videos. More sophisticated and rigorous treatment of security and privacy control is left as future work.

The overview of CrowdV ision is depicted in Fig. 1. A mobile device has several options to perform object detection on each related video. Performing object detection in a video entails two main steps. First, video frames must be extracted from the video and turned into images. Second, object detection is performed on the images. These two functions may be performed locally on a mobile device, or in the cloud, or distributed between the two systems.

As illustrated in Fig. 1, by considering several input parameters and constraints, e.g., network connections, battery life, and data budget, a mobile device can perform

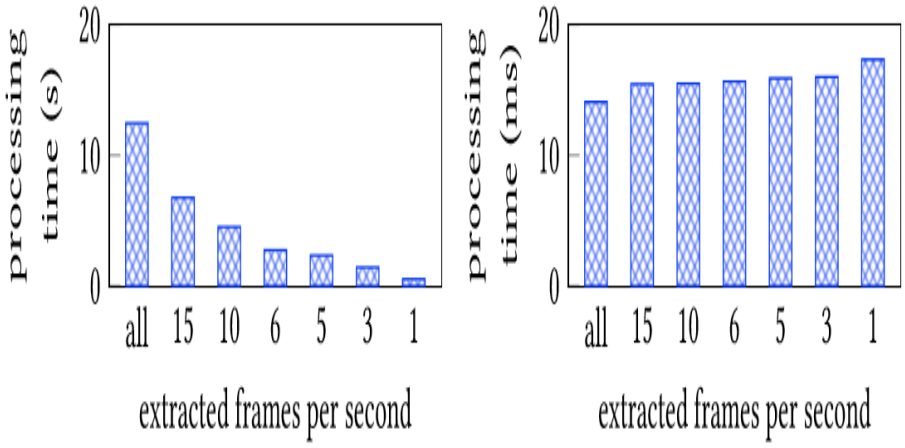


Fig. 3.2: Processing time of different extraction rates on a 30-second 1080P video using DSP on smart phone

(i) video ofoad by sending the whole video to the cloud. Alternatively, the user may perform frame extraction locally and then ofoad specic frames to the cloud. In this case, it needs to determine whether each frame is processed by (ii) frame ofoad in which the frame is sent to the cloud for detection or (iii) local detection where the frame is detected on the mobile device.

In Crowd Vision,we consider both the task issuers query requirements and the resource usages of mobile devices. From the perspective of the task issuer, the quality of CrowdVisions response to the query can be evaluated by two metrics. First is the timeliness of the response, measuring the time required to process all related videos on each mobile device. Second, the accuracy of the response is measured by the frames identied as containing the event that actually contains the event, which is determined by the CNN model. In this work, we consider timeliness. From the users perspective, its primary criteria is the resource efciency. Moreover, although the computational capability of mobile devices continues to improve, video processing using deep learning on mobile devices is still limited and resource intensive. Therefore, the cloud is provided by task issuers to assist in video processing.

When mobile devices perform video processing with ofoading, they may be connected to the cloud via WiFi or cellular networks. In these two circumstances, users may have different priorities in terms of budgeting resources. When using WiFi, users may not care about the resource usage as long as video processing does not affect the normal use of their mobile devices; or they may only care about battery life when battery recharging is not accessible. When using cellular networks, battery life and data usage may become the primary concerns since it is usually inconvenient to recharge mobile device when connected cellular networks and cellular data plans may be limited.Therefore,the design of CrowdVision takes both of these into consideration.

Through the characterization of processing time, energy, and cellular data usage for video processing, we design CrowdVision. By considering inputs from the processing task,constraints of users,and various network scenarios,we optimize cloud-assisted video crowd processing as a function of these design parameters. CrowdVision determines how to quickly and efciently process each video and takes advantage of some actions that can be done in parallel or pipelined. In the following, we describe each of these processing components.

3.1 Frame Extraction

Frame extraction isused to take individual video frames and transform them into images upon which object detection may be performed.To target objects with different dynamics within the video, the task issuer may request a different frame extraction rate. For example, for an object moving at a high speed like a car,the rate should be high enough to not miss the object.For a slowly moving object like a pedestrian, a lower frame extraction rate can be used, which eases the computing burden. The setting of the frame extraction rate for object detection is important but it is not the focus of this paper. CrowdVision takes frame extraction rate as a parameter dened by the task, denoted as er frame per second (fps).

Fig.2a and 2b,respectively,illustrate the total processing time and the processing time per frame for frame extraction using a hardware codec (H.264) with different extraction rates on a 30-second 1080p video (30fps) on a Galaxy S5 (unless stated otherwise all measurements are performed on the Galaxy S5). In Fig. 2a, the total processing time for extracting all the frames takes about 13 seconds, and it decreases as the extraction rate reduces. The processing time per frame, as illustrated in Fig. 2b, slightly and linearly increases as the extraction rate decreases, because it takes longer time to seek a frame when fewer frames are extracted. The average is about 16 ms. Moreover, as shown in Table 1, the power of frame extraction is about 1W, and the CPU usage is only about 4%Ṫhe power and energy is measured by Monsoon power monitor.

3.2 Detection

Detection is the process of determining if the object of interest is in a frame. For detection, we currently use AlexNet, a 8-layer CNN (totally 729M FLOPS), on Caffe to perform classication,but we do not have any restrictionon CNN models1.Although Caffe can be accelerated by CUDA enabled GPUs,it is currently not supported by GPUs on off-the-shelf mobile devices. Caffe in both the cloud and mobile devices is the same, but the cloud is equipped with powerful CUDA-enabled GPUs and can perform object detection hundreds of times faster than mobile devices.

We nd that the processing time of detection is affected by the batch size,e.g.,processing two frames in abatchtakes less time than that of two frames that are detected individually. Fig. 3 shows the measured processing time of detection performed individually and in a batch. The processing time grows linearly with the increase of the number of frames for both cases. However, batch processing performs much better, where the intercept () is about 240ms and the slope () is 400ms. That means if we process ten frames one by one, it takes about 6.4s, while if ten frames are processed in a batch, it takes about 4.24s. The difference grows with the increase of the number of frames. For

detection, it is better to put more frames in a batch to reduce processing time. However, the system must wait longer to get the extracted frames from the video. Therefore, it is difcult to determine the best batch size.

This characteristic of batch processing commonly exists in the computing of CNNs on both CPUs and GPUs, and it also drives the design of CrowdVisions modules of local detection and frame ofoad. Although CrowdVision is designed based on the detection using mobile CPUs, it can be easily adapted to mobile GPUs when they are available

Fig. 3.3: Power and CPU usage of frame extraction and object detection on smart phone

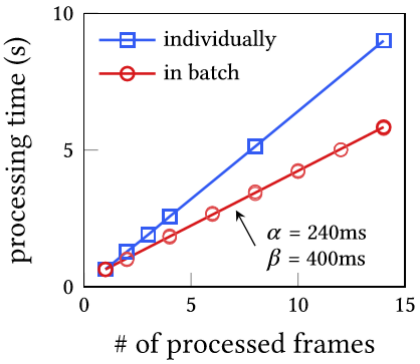


Fig. 3.4: Processing time of detection performed individually or in batch on smart phone

In order to maximally benefit from the abundance of multimedia data and make optimal use of the rapidly developing multimedia technology, automated mechanisms are needed to establish a similarity link from one multimedia item to another one if they are semantically related, regardless of the type of modalities (e.g. text, visual or audio) of the items. Modeling of similarity links has traditionally focused mainly on single-modality scenarios. For instance, information retrieval has focused on bringing similar textual documents together, while content-based image, video or audio retrieval has attempted the same for images, videos and audio items, respectively. In order to provide an answer to the above challenge, research towards reliable solutions for cross-modal applications, e.g. cross-modal retrieval, which operate across modality boundaries, has

gained significant momentum recently. Since features of different modalities usually have inconsistent distribution and representation, a modality gap needs to be bridged, that is, ways need to be found to assess semantic similarity of items across modalities. A common approach to bridge the modality gap is representation learning. The goal there is to find (i.e.,learn) projections of data items from different modalities into a common (modality agnostic) feature representation subspace in which the similarity between them can be assessed directly. A variety of cross-modal retrieval methods have been proposed recently, which propose different ways of learning the common representation subspace.for the acceleration of deep learning on off-the-shelf mobile devices, because this just requires a change of system parameters. Moreover, despite the fact that mobile GPUs can perform several times faster than CPUs, ofoading may still be needed since CNN models go deeper and computing becomes much more difcult (e.g., the state-of-the-art CNN, ResNet , has more than a thousand layers).

Table 1 gives the measured power and CPU usage of performing detection on the smart phone. The power is 2191mW, while the CPU usage is 25% (occupied one of four cores). Although object detection requires considerable computation, together with frame extraction, it will not affect the normal operations of mobile devices, and thus we also consider performing video processing on mobile devices locally.

3.3 Ofoading

Mobile devices can choose between video ofoad and local processing as depicted in Fig. 1. When choosing local processing, they then decide between frame ofoad and local detection. These decisions are all affected by network conditions that lead to divergent throughput and power. Moreover,mobile devices can be connected to the cloud via either WiFi or cellular.Each may have different characteristics,and users may also have different concerns in each scenario. Therefore, we separate these two scenarios throughout the design of CrowdVision for easy of presentation. However, it does not

mean that the solution introduced for one scenario works exclusively in that scenario. It depends on situational characteristics.

Note that due to the characteristic of batch processing, none of existing ofoading schemes can directly and effectively apply to deep learning for video processing because they cannot take advantage of batch processing to reduce the completion time and energy expenditure

Chapter 4

PROCESSING UNDER WIFI

When mobile devices are connected to the cloud with WiFi, energy usage may or may not be the concern. Therefore, in this section, we study the optimization of the processing time on a mobile device, and then we investigate the optimization while also considering energy usage as a constraint, which can be imposed by the user to govern the amount of energy the device uses for each task.

4.1 Optimizing Completion Time

When a mobile device receives a task, it rst parses the task to nd the related videos based on metadata, such as location information, or timestamp. The completion time is the time spent processing these related videos. We say a video is processed when the video is ofoaded to the cloud, or frames are extracted and processed by some combination of the cloud and local device. In CrowdVision, multiple videos are processed in serial; i.e., we do not consider parallel video ofoading and local processing, since it is not resource-efcient. For example, given local processing and video ofoading, one of them is more energy-efcient. Parallelizing video ofoading and local processing will cost more energy than selecting the more energy-efcient way and process videos in serial. As videos are processed in serial, optimizing the completion time of all related videos on a mobile device is equal to minimizing the completion time of each video. Therefore, for each video, CrowdVision rst estimates and compares the completion time of video ofoad and local processing

Mobile devices are usually connected to WiFi when users are at home or at work. In such environments, the channel conditions commonly vary slightly. Therefore, similar to MAUI, for the processing under WiFi, the data rate between a mobile device and cloud can be considered to be stable duringa short period time.Note that the environment with a highly dynamic WiFi data rate can be handled by the adaptive algorithm discussed in5. Like MAUI, mobile devices probe the data rate by sending 10KB data to the cloud periodically. Let r denote the data rate.

For each video, the processing time of video ofoad can be easily estimated. However, for local processing, we have to consider the time spent on extracting a frame, ofoading a frame, and detecting a frame, and then choose between frame ofoad and local detection wisely for each frame to obtain minimal completion time of local processing.

Processing time on frame extraction. As shown in Fig. 2b, the processing time of extracting a frame linearly increases as extraction rate decreases.Given an extraction rate dened by the task, we can easily obtain the processing time to extract a frame from a specic video. Note that videos with different resolutions (e.g., 720p, 1080p, 4K) have varied processing times due to different decoding workload.However, the processing time on the frames from videos with the same specications varies only slightly based on our experiments. Processing time on local detection. As illustrated in Fig. 3, the processing time of frame detection can be calculated as + x, where x is the number of frames included in a processing batch. However, since extracted frames do not become available at the same time, and batch processing requires all the frames to be processed to be available before processing,the optimized processing time of multiple extracted frames cannot be simply computed as above. This problem turns out to be non-trivial.

Assume there will be totally n frames extracted from a video. Each frame will be available at a time interval (i.e., the time to extract a frame). To minimize the processing time, we need to optimally determine how to process these frames, i.e., how many

processing batches are needed and how many frames each batch should process. To mathematically formulate the problem, let us assume the number of batches is also n (n can be seen as the maximum number of batches needed to process the frames) and let yi denote the number of frames for batch i [1, n],where yi 0 (yi = 0 means batch i does not process any frame). Let xi denote the time interval between the start times of batch i and i+1, x0 denote the waiting time before processing the rst batch, xn denote the processing time of the last batch. Note that if yi and yi+1 are both zero, xi is also zero.Then,our problem can be formulated as an Integer Linear Programming (ILP) problem as following:

$$\min \quad \sum_{i=0}^{n} x_i \tag{1}$$

$$s.t. \quad \sum_{i=1}^{n} y_i = n, \tag{2}$$

$$\sum_{i=0}^{k} x_i \geq \sum_{i=1}^{k+1} \gamma y_i, \qquad k = 0, \ldots, n-1 \tag{3}$$

$$x_i + b_i \alpha \geq \alpha + \beta y_i, \qquad i = 1, \ldots, n \tag{4}$$

$$n(1 - b_i) \geq y_i \geq 0, \qquad i = 1, \ldots, n \tag{5}$$

$$b_i = \{0, 1\}, \qquad i = 1, \ldots, n. \tag{6}$$

Fig. 4.1: Equations

Constraint (2) regulates that all frames are processed. Constraint (3) makes sure that the number of frames to be processed by a batch are available before the start of the batch. Moreover, the previous batch must have been completed already before processing a batch, which can be represented as xi + yi if yi ¿ 0 and else xi 0. Constraints (4)(5)(6) are the workaround for this. If yi ¿ 0, bi is zero according to constraint (5). If yi = 0, constraint (4) is equal to xi+bi .As our problem is a minimization problem,b will be equal to one. Since the problem (1) is ILP (NP-hard), the optimal solution of minimizing the processing time on detection costs too much, even for a small n. For example, for a instance of n = 100 with parameters obtained from Fig. 2 and 3, GLPK takes 217 seconds to solve the problem optimally on a 16-core workstation.

Processing time on frame ofoad. Besides detecting frames locally, mobile devices can also choose to ofoad frames to the cloud to accelerate the processing. Let denote the time a mobile device spends ofoading a frame, where = df/r and df is the data size of an extracted frame. Note that the extracted frame may be resized to feed different CNN models and thus df is not a xed size. When ofoading a frame takes less time than extracting a frame (i.e.,), the completion time of local processing is about n, which is the processing time of frame extraction. However, when ¿ (the most common case), there will be frame backlog and local detection is needed. However, as discussed above, minimizing the processing time of local detection is already an NP-hard problem.The problem that considers both frame ofoading and local detection to minimize the processing time of local processing is even more difcult to solve. Therefore, we propose a split-shift algorithm to solve the problem.

4.2 Split-Shift Algorithm

For each extracted frame, we have two options: frame ofoad and local detection, which are two processes working in parallel to reduce the completion time of a video. Intuitively,the ofoading process should keeps ending extracted frames to the cloud. For the detection process, it is better to reduce the number of processing batches (i.e., increase batch sizes), since each additional batch incurs more processing time. However, as the batch processing requires the input frames be available before the processing starts, it is better to not wait too long for the extracted frames. Based on this intuition, we design the split-shift algorithm.

If only frame ofoad is deployed, the completion time is n (accurately it is n + , but is small and cannot be reduced and thus we consider n for simplicity). We treat thedetectionprocessasahelpertoreducen.The main idea is to shift the workload from the ofoading process to the detection process to balance the completion times of these two processes by determining the number of processing batches and the number of frames to be pro-

cessed in each batch.

First, let us assume that all frames are available at the beginning. Then, let n p denote the number of frames to be detected locally that minimizes the completion time,and we have

$$\delta(n - n_{\boldsymbol{p}}^{*}) = \alpha + \beta n_{\boldsymbol{p}}^{*},$$

$$\delta(n - n_p) = n_p\gamma' + \alpha + \beta n_p,$$

$$n_p^1\gamma' + \alpha + \beta n_p^1 \geq n_p^*\gamma'$$

The work ow of the split-shift algorithm is illustrated in Fig. 4. The computational complexity of the algorithm is O(n), which is desirable for mobile devices. Moreover, it is also easy to be implemented on mobile devices; i.e., the ofoading process simply keeps sending frames one by one until frame queue is empty, while the detection process initiates batch processing when the number of frames required by each batch are available in the frame queue. Overall, for each video, a mobile device rst estimates the completion time of video ofoad and local processing, respectively,and then chooses the approach that has a better completion time.

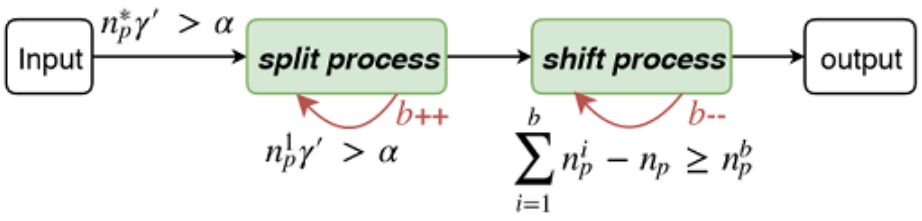


Fig. 4.2: Equations

Chapter 5

PROCESSING UNDER CELLULAR

When mobile devices have only cellular connections, it is better to not ofoad videos, due to the limitation of cellular uplink speed and limitations on data usage. Since data transmission rates may vary widely over time, e.g., due to movement, the solution for processing under WiFi may not be valid for the cellular scenario.

Obviously, if all frames are detected locally, there is no cellular data usage. However, this may consume signicant amounts of energy and severely increase completion time. Although users may be more sensitive about data usage rather than energy, energy usage is a concern. Data usage can be easily controlled, while the energy cost of frame ofoad varies with data transmission rate and signal strength, and thus it is hard to tell how much energy will be consumed to ofoad a frame beforehand.

Therefore, for processing under cellular, we consider the problem of optimizing both completion time and energy consumption with a data usage constraint so that decisions are made while considering trade offs between these two objectives. However, due to the variation of cellular data rates, we cannot solve the problem traditionally by Pareto optimal solutions. Thus, we design an adaptive algorithm that makes a decision on each extracted frame (i.e., between frame ofoad and local detection) towards optimizing both objectives.

To perform video processing under cellular, users must specify a data usage constraint D0 between zero and a maximum, which can be easily calculated, i.e., video duration frame extraction rate frame data size. If D0 is equal to zero, the problem is trivial and all extracted frames are detected locally. If the user does not care data usage, D0 is simply set to the maximum. In the following, we consider the case that D0 is greater than 0.

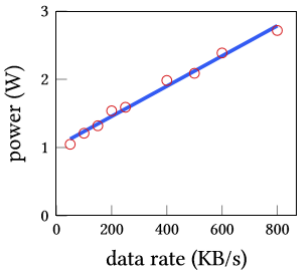


Fig. 5.1: Power of LTE in terms of different uplink rates on smartphone.

5.1 Estimation of Uplink Rate and Power

Before deciding between detection and ofoading for each frame, we need to know the cost in terms of processing time and energy. For frame detection, both processing time and power are stable and can be accurately estimated, although the processing time varies with the number of processing batches.The difcultylies in estimating these for frame ofoad. The ofoading time and power are related to many factors, such as signal strength, channel conditions and network trafc. However, from measurements on the smartphone, we can see the power of LTE uplink exhibits an approximately linear relationship with data rates as depicted in Fig. 6. Similar results are also found. Therefore, during a short period time, we can explore the history of previous frame ofoads to correlate data rates and power. Moreover, among the factors that affect cellular uplink rates, signal strength, which is mainly impacted by the users location and movement, can be treated as an indicator of data rate during a short period of time, where the coefcients of the linear relation between data rates and power are steady.

To estimate the uplink data rate and power, rst we record the data rate and energy consumption for each ofoaded frame. Then, these records can be exploited to derive the linear relation between data rate and powerusing regression, to maintain an up-to-date correlation estimate. Moreover, we also record the cellular signal strength level during each frame ofoad. Since generally data rate has a linear relation ship with signal strength during short periods of time, we can exploit current signal strength level to roughly

estimate current data rate based on the records of previously ofoaded frames.Then,the estimated data rate is exploited to gauge energy consumption to ofoad a frame.

5.2 Adaptive Algorithm

As aforementioned, we try to optimize completion time and energy consumption together. However, due to the dynamics of cellular uplink data rate and power,we propose a tailored solution for this specic problem rather than a scalar treatment of these two objectives.

For the processing under cellular, we have two options: frame ofoad and local detection. Frame ofoad may improve completion time or energy consumption or both, depending on the cellular data rate and power consumed by cellular during ofoading. For a number of frames, local detection can be exploited to reduce the completion time by increasing the number of processing batches, although this incurs an additional energy cost. Moreover, local detection usually takes multiple frames as input, and once it starts processing, the frames should not to be ofoaded to avoid duplicate processing. Therefore, it is difcult to determine the number of frames included in batch processing, since ofoading frames may improve performance during batch processing.

Frame ofoad may be exploited to reduce both completion time and energy consumption simultaneously; local detection cannot optimize these two objectives together,and hence it may be preferred only when both completion time and energy cannot benet from frame ofoad. In addition, unlike the processing under WiFi, we do not parallelize frame ofoad and local detection,since this always sacrices one objective for another. Based these considerations, we design an adaptive algorithm for the processing under cellular, towards optimizing both completion time and energy cost with the cellular data usage constraint, which works as follows.

Frames are continuously extracted from a video into a frame queue. When a frame is available, a mobile device will ofoad a frame to the cloud. If both the ofoading time and energy consumption are less than those estimated for local detection,this ofoading is called a successful attempt; otherwise, it is referred to as an unsuccessful attempt

After the rst successful attempt, the mobile device will ofoad another frame. If this is also a successful attempt, then we can derive the linear relationship between data rate and energy consumption, and the relation between signal strength and data rate. For subsequent ofoads, the mobile device can estimate the completion time and energy consumption based on current signal strength and the linear functions. If both are less than those estimated for local detection, it will ofoad another frame.

For local detection, each time the mobile device will process as many frames from the frame queue (it may wait for frames to be extracted from a video), which can be completed within .After a time out of the back off timer,the mobile device will switch back to frame ofoad.The process iterates until frame ofoad reaches the limit nd or all frames are processed. If limit is reached, then local detection will be the only option to process the remaining frames. Frame ofoad and local detection are performed alternately to nd the moments when frame ofoad can improve both energy and completion time, or when local detection should be performed. The back off timer exponentially increases with thenumberofconsecutiveunsuccessful attempts. This is designed to capture different network conditions. When the network condition is constantly poor, ofoading is less frequently attempted and the frequency is exponentially reduced. When the network condition varies,ofoading is attempted more frequently to seize the moments when frame ofoad benets both.

Insteadofpa ra llelizingfram eofo adandl oca ldete ction, we choose to perform them alternatively. Frame ofoad is employed to optimize both completion time and energy consumption.However,duetoth ev ari ationofne tw orkcond itions, frame ofoad is se-

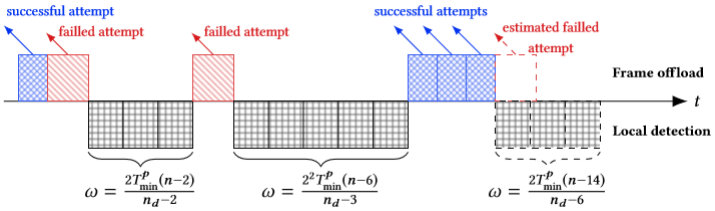


Fig. 5.2: Illustration of video processing under cellular.

lected only if it outperforms local detection in terms of both completion time and energy and meets the data usage constraint. The adaptive algorithm is simple and efcient, and it can also be easily implemented.

Moreover, under cellular, we do not consider video ofoadingandi tim pl icitlymean sthed ata size ofal le xtr acted frames of a video is smaller than the video itself. Thus, the required storage for the adaptive algorithm is also the size of the largest video related to the query.

Chapter 6

IMPLEMENTATION

We have implemented CrowdVision on Android and a workstat ionwith aGTX T ITA NXGPUa sth ec lou dtoas si st mobile devices for video crowdproc essing. Tasks are issued from the workstation to mobile devices through Google CloudMess aging.Messagingbetw eenthec lou dandm obi le devices are implemented using Protocol Buffers. The architectureandA ppo fCr ow dVisionarei llu stratedinFi g. 8.The implementation of CrowdVision on mobile devices consists of three components: core services, monitors, and a GUI. Core Services. Core services contain an executor service, a frame extraction callable, a Caffe callable, an ofoading callable, a multithre aded frame queue, and a video database. The video database stores the metadata of local videos such that CrowdVision can quickly screen videos for the processing task. The Multithre aded frame queue stores extracted frames from videos and supports multithre ading access.The executorserv ce isable tocall anyof t hecall ables to perform frame extraction, Caffe for frame detection, or ofoading of a frame or video. The executor service takes inputs from tasks, GUI and monitors, and employs the selected strategy to process each video.

Monitors. There are two monitors to measure network state and battery level, which provide inputs to the core services. The network monitor tells the cores services current network connection with distinct metrics for WiFi or cellular networks. For WiFi, it measures the uplink data rate from a mobile device to cloud using probing data. For cellular, it monitors the signal strength level. The battery monitor measures the energy consumption for each ofoaded frame for the cellular case.GUI: The GUI allows mobile users to congure the energy usage, cellular data usage, and access to videos. Energy control ranges from the minimum to the maximum energy cost of local video

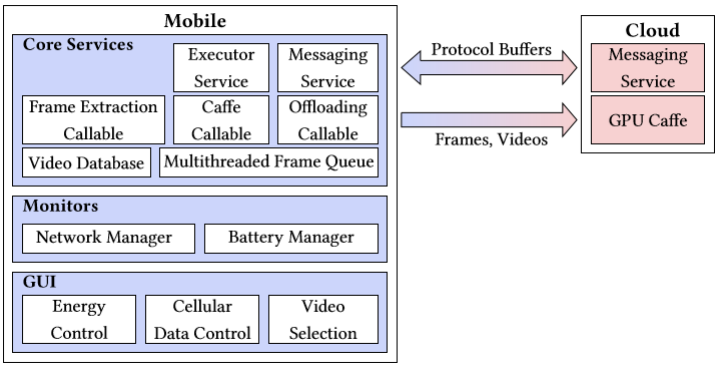


Fig. 6.1: Architecture and App of CrowdVision.

processing. It is enabled using WiFi.

Chapter 7

EVALUATION

In this section, we evaluate the performance of CrowdVision. We rst compare CrowdVision against alternatives under various system settings based on empirically gathered measurements to understand when and why CrowdVision outperforms alternatives and then conrm the gains of CrowdVision through experiments on our testbed. We also investigate how the split-shift algorithm approximates the optimum for determining processing batches.

7.1 Performance under Various Settings

We rst use empirically gathered measurements of processing time and energy taken from a Galaxy S5 to investigate the impact of system parameters, such as WiFi/cellular data rate, frame data size and frame extraction rate. These parameters are set to correspond their values in real applications. Under WiFi, CrowdVision is compared against MAUI [10] (the most popular generic computation ofoad system). Since MAUI chooses to ofoad all computations with default settings to preserve energy.Therefore ,we adapt MAUI to optimize the completion time rather than energy. Under cellular, MAUI has to repetitively recalculate the solver at run time to adapt to varying data rate .However,the MAUI solver is not easy and indeed an ILP problem. Since the number of frames is large, the MAUI solver itself takes long time to complete and thus cannot be easily adapted to the scenario under cellular, where the transmission rate dynamically changes and the ofoad decision need to be made at runtime . Therefore, CrowdVision is compared against basic processing options (i.e .,solely frame ofoad and solely local detection). The simulation is carried out on a 1080p 30 second video.Note that the video

specication and duration do not affect their relative performance.

WiFi. Fig. 9a illustrates the completion time of MAUI and CrowdVision in terms of WiFi data rate, where the frame data size df is 500KB and frame extraction rate re is 6fps. When the WiFi data rate is high enough, e.g., 2620KB/s in Fig. 9a, video ofoad costs the least and both MAUI and CrowdVision choose video ofoad and hence perform the same. Similarly, when the WiFi data rate is low enough, local detection will be selected by both MAUI and CrowdVisionandt hus they perform equivalently again,e.g., 20KB/s in Fig. 9a. Note that when the WiFi data rate is less than 1000KB, the completion time of MAUI is the same, because under these WiFi data rates MAUI chooses local detection (no ofoading). When between these two rates, CrowdVision outperforms MAUI. This is because the split shift algorithm employed by CrowdVision can improve completion time by taking advantage of batch processing and by parallelizing frame ofoad and local detection. The completion time of CrowdVision is as low as 60% of MAUI as depicted in Fig. 9a.

Since CNN models may require different resolutions of images as input, which depends how a CNN is architected, we also investigate the effect of frame data size on the completion time. As illustrated in Fig. 9b, when frame data size is small, frame ofoad is the best option. When frame data size is large enough, it is better to ofoad videos since the WiFi data rate is high 2000KB/s. In both regions, MAUI and CrowdVision perform equally.However,between them, CrowdVision always outperforms MAUI as depicted in Fig. 9b due to the same reason discussed above. To detect different objects, different frame extraction rates may be dened by the task issuer. CrowdVision still outperforms MAUI as shown in Fig. 9c. Note that when the frame extraction rate is high, both CrowdVision and MAUI perform video ofoad, since only the completion time of video ofoad is not impacted by the frame extraction rate.

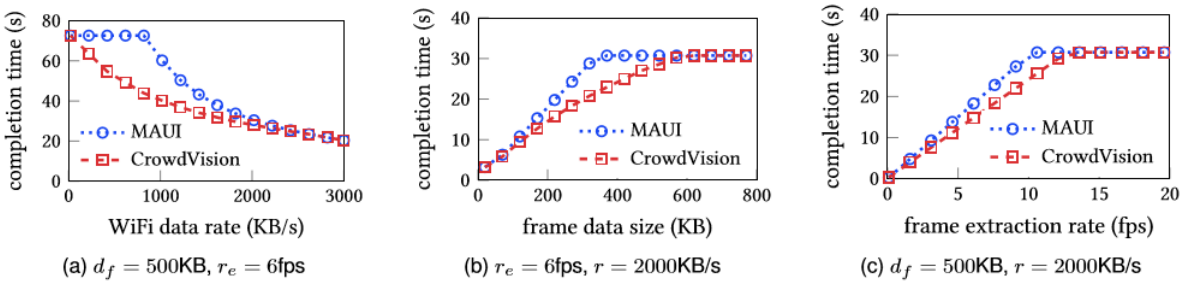


Fig. 7.1: Performance of CrowdVision and MAUI under WiFi in terms of WiFi data rate, frame data size, and frame extraction rate.

Chapter 8

Discussion

Crowd processing.Unlike crowd sourcing or crowd senssing we define crowd processing as a computing-oriented approach.Instead of simply sensing and sharing data,we focus on exploring the computing capability of mobile devices. For crowd processing, the key problem, which is also the focus of this paper,is to enable resource-constrained mobile devices to efciently perform such complex computing. In this paper, we consider to optimize the completion time with/without energy constraint and optimzie them together. For other use cases, it is indeed important to preserve battery and reduce local computation on users devices. Enabling CrowdVision for other use cases involves optimizing different metrics and needs a set of carefully designed solutions and will be considered in our further work.

Scalability. When there are a large number of users who have the video related to the query, the bandwidth of the cloud seems the bottleneck if most users choose to ofoad computation. However, as the data rate between mobile device and cloud decreases, the mobile device will in turn perform more computation locally. Moreover, we argue that the provision of computational capability of the cloud is the responsibility of the task issuer. Each individual device makes ofoading decisions without considering the computational capability of the cloud. Therefore, our system has good scalability.

Compatibility. CrowdVision is currently built on top of Caffe. However, as shown in Fig. 8, Caffe works only as a callable to perform detection on mobile device and cloud. Therefore, it can be easily replaced by other deep learning frameworks. Moreover, CrowdVision can be easily adapted to mobile GPUs when they are available for

the acceleration of deep learning on off-the-shelf mobile devices,because it is just the change of system parameters. It is worth noting that the computing capability of mobile devices even equipped with mobile GPUs is still far behind GPU-accelerated cloud. Generality. The characteristic of batch processing commonly exists in the computing of deep learning, not just for convolutional neural networks. Therefore, CrowdVision that is particularly designed to take advantage of batch processing can be used for other applications with minor modications. Moreover, the split-shift algorithm can be exploited to determine batch processing so as to optimize the performance and resource usage in these applications with/without computation ofoad.

Chapter 9

Conclusion

In this paper, we present CrowdVision, a computing platform for crowd processing videos using deep learning.CrowdVision is designed to optimize the performance on mobile devices with computational ofoad and by taking into consideration the characteristics of the computing of deep learning for video processing. CrowdVision is implemented and evaluated on the off-the-shelf smart phone. Experimental results demonstrate that CrowdVision greatly outperforms the existing computational ofoad system or basic processing options under various settings and network conditions. We envision CrowdVision to be a great computing frame work for crowd processing applications using deep learning.

References

[1] Z.Lu,S.Ralla palli,K.Chan,and T.La Porta Modeling the resource requirements of convolutional neural networks on mobile devices, in MM17.

[2] X.D.Yang,G.Xue,X.Fang,and J.Tang,Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing,in MobiCom12

[3] D.Geiger,M.Rosemann,E.Fielt,andM.Schader,Crowdsourcing information systems: denition,typology and design,in Proceedings of the 33rd International Conference on Information Systems. Association for Information Systems/AIS Electronic Library (AISeL), 2012.

[4] B.Morschheuser,J.Hamari,and J.Koivisto,Gamication in crowdsourcing: a review,in HICSS,2016,pp.43754384.

[5] D.P.Anderson,Boinc:A system for public-resource computing and storage, in proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, 2004, pp. 410.

[6] M.-R.Ra,B.Liu,T.L.Porta,and R.Govindan,Medusa: A programming framework for crowd-sensing applications, in MobiSys12.

[7] Y.-H.Kao,B.Krishnamachari,M.R.Ra,Bai Hermes:Latency optimal task assignment for resource constrained mobile computing,IEEETransactions on MobileComputing,vol.16,no.11, pp. 30563069, 2017.

[8] C. You,K.Huang,H.Chae,and B.-H. Kim,Energy-efcient resource allocation for mobile-edge computation ofoading,IEEE Transactions on Wireless Communications, vol. 16, no. 3, pp. 1397 1411, 2017.