Poster: Approximate Caching for Mobile Image Recognition

James Mariani, Yongqi Han, Li Xiao
Department of Computer Science
Michigan State University
East Lansing, United States of America
mariani4@msu.edu, hanyongq@msu.edu, lxiao@cse.msu.edu

***Abstract*—Many emerging mobile applications rely heavily upon image recognition of both static images and live video streams. Image recognition is commonly achieved using deep neural networks (DNNs) which can achieve high accuracy but also incur significant computation latency and energy consumption on resource-constrained smartphones. We introduce an in-memory caching paradigm that supports infrastructure-less collaborative computation reuse in smartphone image recognition.We propose using the inertial movement of smartphones, the locality inherent in video streams, as well as information from nearby, peer-to-peer devices to maximize the computation reuse opportunities in mobile image recognition. Experimental results show that our system lowers the average latency of standard mobile neural network image recognition applications by up to 94% with minimal loss of recognition accuracy.**

***Keywords*-Image Recognition, Approximate Caching, Device-to-device Communication**

I. Introduction

Emerging mobile applications are more focused on technology interacting with, and augmenting the real-world environment the user occupies. This interaction with the real-world relies heavily on image recognition, or the ability for a smartphone to be able to determine what it is looking at through its camera. For example, augmented reality applications span navigation [6], gaming [1], education [7], etc. The main issue plaguing mobile augmented reality is the computational intensity of running large neural networks on resource constrained smartphones.

Most techniques for mitigating the latency and computation restrictions of smartphones include offloading to cloud or edge servers [5], reducing the complexity of the deep neural networks (DNNs) often used for mobile image recognition [2], and caching results for reuse [3].

We propose a system built on in-memory approximate caching to reduce the latency of mobile image recognition to an acceptable level for a seamless user experience. In this paper we propose an approximate caching system that introduces caching strategies built around the inertial movement of mobile devices and allows for infrastructure-less collaboration between nearby devices through creation of ad-hoc peer-to-peer networks. In contrast to current work, our solution optimizes cache management using the inherent mobility and collaborative nature of smartphones. We offer a suite of inertial-driven optimization tools to both reduce the latency of image recognition and maintain high accuracy.

II. System Design

A. Overview

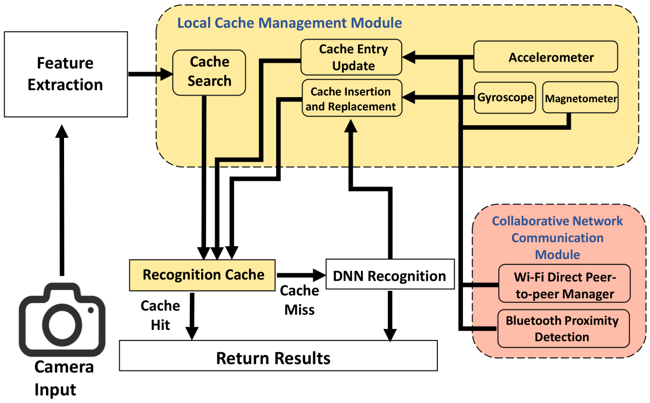
Local Cache Management Module
Feature Extraction
Cache Search
Cache Entry Update
Cache Insertion and Replacement
Accelerometer
Gyroscope
Magnetometer
Collaborative Network Communication Module
Wi-Fi Direct Peer-to-peer Manager
Bluetooth Proximity Detection
Recognition Cache
DNN Recognition
Cache Hit
Cache Miss
Return Results
Camera Input

Figure 1. System Architecture

Our system is an in-memory approximate caching system that facilitates the reuse of image recognition computations to improve the overall latency and efficiency of DNN image recognition on mobile devices. We leverage the collaborative nature and inertial movement of mobile devices, as well as the temporal, spatial, and semantic locality of real-world video feeds, to both optimize cache searching and also ensure that the correct computations are cached for reuse. Approximate caching uses image similarity instead of exact matches to determine cache hits or misses, which is perfect for real-world situations with video streams. Our approximate cache is built on a Locality Sensitive Hashing data structure.

The core of our contributions lie in the Local Cache Management and Collaborative Network Communication Modules, seen in Figure 1. The Local Cache Management Module handles all cache searching, cache insertion and replacement, sampling the on-device accelerometer, gyroscope, and magnetometer, and updating the cache entry reuse scores. The Collaborative Network Communication Module houses both the Wi-Fi Direct and Bluetooth components of our system. This includes the Bluetooth proximity detection used for sensing the relative movement between users, and the Wi-Fi Direct peer-to-peer logic for actually sending over cache entries based on the Bluetooth proximity readings.

B. Caching Optimizations

To optimize our cache management, we introduce many inertial-driven techniques that make use of a smartphone's

This work was partially supported by the US National Science Foundation (NSF) under grant CCF-2007159

on-board inertial measurement unit. We better inform our cache replacement algorithms with inertial data from a user's device to make accurate guesses about what objects might currently be in the view of the camera based on how the user has moved through space while recognizing objects. Additionally, we determine the rotation of a user's smartphone and use this information to cache more effectively, as our initial experimentation has shown that this rotation is indicative of certain cache patterns.

Additionally, we use ad-hoc WiFi to connect nearby users together who then share information about known objects in the user's vicinity. We augment our device-to-device WiFi network with proximity detection using Bluetooth's discovery feature. Determining the proximity of a user in relation to other users helps us weight the object recommendations we've received, based on how close a neighbor is to us physically. Users in our system share only cache entries with each other. Cache entries only contain the extracted features of an image as well as the image's label. It is not possible to reconstruct a raw image based only on the features extracted from that image, so we do not foresee any security issues with our system.

III. Preliminary Implementation and Evaluation

A. *Evaluation Methodology*

To test the viability of our design, we develop an image recognition Android application built upon various popular DNN models. The application was developed using Java, OpenCV, and TensorFlow to facilitate the usage of multiple DNNs.

We developed a prototype that we test on five different smartphones, ranging in age, cost, and computational capability. We compare against a baseline of raw neural network recognition with no optimizations. We use the ResNet [4] and Inception v4 [8] models for testing and average the results.

We supports both local caching and collaborative caching. Figure 2 shows the latency reduction achieved by our in cases from one to four users. With only one device, we achieve latency reduction of anywhere from 80% to 84% depending on what device is being used. With two devices collaborating, we achieve latency reduction anywhere from 87% to 90%. With three users we see latency reduction in the range of 89% to 92%. Finally, with four devices we achieve latency reduction anywhere from 91% to 94%. To achieve these results, our cache has a reuse rate of 94% and an error rate of 12%.

These results are significant in two areas. First, when there are many neighbors nearby, we can achieve very small latencies, and even in situations when there are no neighbors, our system still achieves significant latency reduction, which is important as it is not always viable to have neighbors nearby to communicate with.

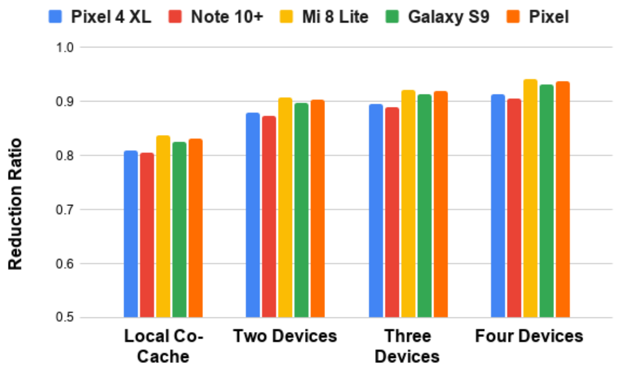

Figure 2. Latency Reduction with Multiple Users

IV. Conclusion

In this paper we introduce an inertial-driven collaborative approximate caching system where computation results of similar images can be reused to improve the latency of mobile image recognition. Approximate caching enhanced with our optimizations can achieve low latency without sacrificing overall accuracy. We build a prototype system and evaluate its effectiveness in a realistic real-world situation. Our evaluation shows that our system can reduce the overall latency of image recognition on smartphones by up to 94%.

References

[1] Pokemon go augmented reality game, 2016.

[2] Biyi Fang, Xiao Zeng, and Mi Zhang. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 115–127, 2018.

[3] Peizhen Guo, Bo Hu, Rui Li, and Wenjun Hu. Foggycache: Cross-device approximate computation reuse. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 19–34, 2018.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*, MobiCom '19, New York, NY, USA, 2019. Association for Computing Machinery.

[6] Wolfgang Narzt, Gustav Pomberger, Alois Ferscha, Dieter Kolb, Reiner Müller, Jan Wieghardt, Horst Hörtner, and Christopher Lindinger. Augmented reality navigation systems. *Universal Access in the Information Society*, 4(3):177–187, 2006.

[7] Iulian Radu. Augmented reality in education: a meta-review and cross-media analysis. *Personal and Ubiquitous Computing*, 18(6):1533–1543, 2014.

[8] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.