University of Glasgow | School of Computing Science

Social Distancing Detector with Deep Learning and Parallel Computing on Android Application

Jinkawin Phongpawarit

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

A dissertation presented in part fulfilment of the requirements of the Degree of Master of Science at The University of Glasgow

Date of submission placed here

Abstract

abstract goes here

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: ____________________ Signature: ____________________

Acknowledgements

acknowledgements go here

Contents

Chapter 1

Introduction

This report discusses. . .

1.1 Motivation

Test citation [1]. - Why I choose social distancing detection – because Technology helps (become an important role) human in medical problems

- Because technology, development, and research (not only computer, but also med/news/media), it was not cost damage/life/time as it should be. - Example, we have face pandemic for many times - including this year – COVID - And we know how to basically handle this problem - we have found that social distancing is one way to reduce spread [https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/social-distancing.html] - because, Social distancing has been recommended for many years (h1n1) [http://tinyurl.com/ycdxmxcl] - Because Slow down and reduce peak graph [http://tinyurl.com/y94f37qg] - And we still developing and researching for this pandemic - It would be better if technology will help this problem in the future. It will gain advantages - Polices have to work hard to prevent people breaching the law during lockdown - Police or who are involving may be easier (if there is help with technology) - This project may not be the big step to help the world, but I want to be a part of improvement of technology (to help people) - Why I choose mobile application - Mobile become popular device which people use for everything (such as receiving news, study or business purpose) - 1. Performance - One technology that has high competition -¿ high improvement is mobile - Since the first smartphone, it has been evolved a lot - It increases capability of phone (CPU, GPU, and Memory) which is able to perform many tasks as desktop computer - 2. Portability - No charger is needed during being used - Mobile have all needed function (camera, computation hardware - CPU) - Move computation part from server to device - 3. Can be enhance by parallel computing

1.2 Objectives

- Android application is able to do social distancing detection by using Deep Neural Network (DNN) - Able to do the task in parallel - Use camera to detect in the real-time

1.3 Structure

1. Chapter 2…
2. Chapter 3…
3. Chapter 4…
4. Chapter 5…
5. Chapter 6…
6. Chapter 7…

Chapter 2

Background

Introduce background chapter. . .

2.1 Mobile Technology

- What is Android - An open source OS, support mobiles, tablets, watches, TVs, Cars' system [http://tinyurl.com/yag8kyst] - Android development - IDE -¿ Android studio which provides SDKs and tools - Native application -¿ Java and Kolin - Native language -¿ C and C++ 1. For intensive application which require extra performance 2. Accessing native libs (C and C++) - Java and C++ communicate by using Java Native Interface (JNI) [https://developer.android.com/ndk/guides] - Server (and Cloud) is replaced by Mobile Computing [http://tinyurl.com/y6vusamh] - Mobile edge cloud [http://tinyurl.com/y7umteqg]

2.2 Detection Algorithm

- People Detection - Using DNN - What is DNN - How it works - How it works in this project - Insert diagram of DNN - Distance Calculation - Libs - Models

2.3 Parallel Computing

- What is parallel computing - Why I need it - How it works - Insert general picture about Parallel computing - Why it benefits for Android

2.4 Specification

- Samsung S10+ - Android 10 - API Level 29 - 8 Cores - 2 ARM Cortex-A75 2.73GHz - 4 ARM Cortex-A55 1.95GHz - 2 Samsung Exynos M4 1.95 GHz - RAM 8 GB

2.5 Existing Applications

1. Object Detector - 250-300 ms per frame - Live camera 2. Computer Vision Detection - Don't know about (ms per frame) or (frame per second) - Live camera – not smooth - Lots of features including face detection - Problem is it still delay

Chapter 3

Design

3.1 System Architecture

According to Figure 3.1

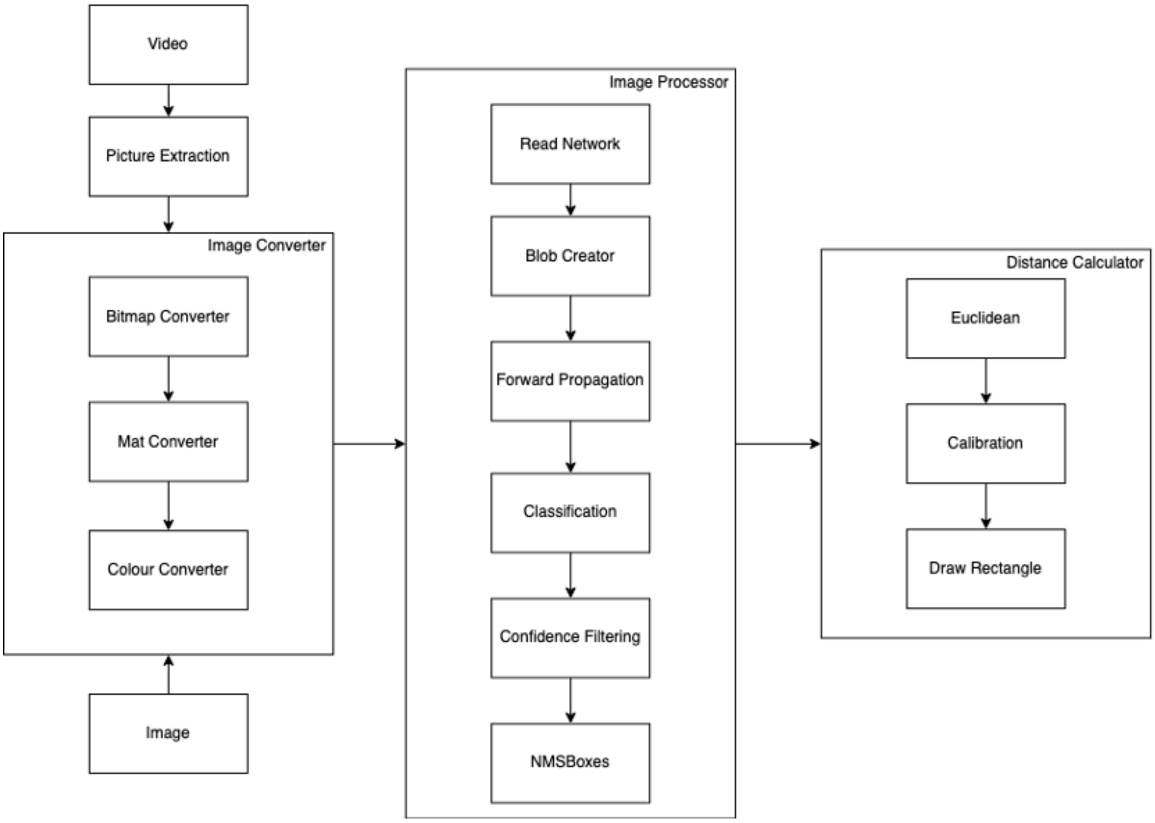

Figure 3.1: System Overview Diagram

3.2 Data Flow Diagram

3.3 Interface Design

Chapter 4

Implement

4.1 Introduction

- System Overview – What I am going to use and implement - This project is implemented on Android Operating System, and Java is used as programming language - Implementation is divided into 3 layers - Java - Process by using CPU - Multithreading by using CPU - Interact and handle evert from user - Manage resource (Ex. Video files, Model Files) - I/O (Ex. Camera, read file, write File) - Permission (Internal and External Storage, Camera) - Each method has to call Native Lib (OpenCV) via JNI

- Java Native Interface (JNI) - Perform as intermediate connection between Java and C++ - JNI is developed by using Native Development Kit (NDK) - And NDK will compile C or C++ into native library by using CMake - C++ - Process by using GPU - Multithreading by using GPU - Directly access to memory - Able to Call OpenCV and other libs without JNI

4.2 Deep Neutral Network

- There are 2 model which are used for doing forward propagation - YOLO3 Model - Mobilenet SSD Model

- How DNN is implemented - DNN is implemented by using OpenCV, and there are steps of processing - Video -¿ Image -¿ Mat - blobFromImage - setInput - net.forward (forward propagation) - determine classification and confidence(accuracy) - NMSBox - Calculate Distance

4.3 Deep Neutral Network

- Intro about parallelisation – How it works in Android - Thread vs ThreadPool - Handler - Multithreading and Multicore - System overview (Manager – Task – Runnable) - ¡Insert diagram¿ - 1 frame per 1 thread - ¡Insert diagram¿

4.4 Parallelisation

- Intro about parallelisation – How it works in Android - Thread vs ThreadPool - Handler - Multithreading and Multicore - System overview (Manager – Task – Runnable) - ¡Insert diagram¿ - 1 frame per 1 thread - ¡Insert diagram¿

4.4.1 Multitheading with CPU

- How to implement - Using Java - Thread pool ¡insert sample of code¿ - Memory Management - Singleton Pattern - Static block - Executed only once - Queue and recycling

Chapter 5

Testing and Evaluation

5.1 Performace Evaluation

- Sequential vs Parallel - 31 frames process vs 16 frames process - Caffe MobileNet SSD vs Darknet YOLO model - YOLO - use more memory because it calls lots of native libs (C++) which is very expensive. - GC collect very often - Programme is frozen - More accuracy - Able to detect person with confidence threshold 0.5 - SSD - Use less memory - No GC collecting - Less accuracy - Able to detect person with confidence threshold 0.3

Model	YOLO		SSD	
Size	960×540	540x480	960×540	540x480
Total Process Time (second)	4.235	3.827	0.337	0.323
Forward Propagation per frame (second)	3.456	3.019	0.284	0.278
Forward Propagation per frame (perenctage)	81.61%	78.89%	84.27%	86.07%

Table 5.1: Picture Processing Performace

Model	YOLO					
	Sequential Computing	Parallel Computing				
		1 Thread	2 Threads	4 Threads	6 Threads	8 Threads
Total Process Time (second)	102.972	117.805	96.415	92.242	88.688	99.441
Garbage Collector (second)	-	0.102	0.280	2.024	3.625	11.333
Process Time without GC	-	117.703	96.136	90.218	85.065	88.108
Forward Propagation (Total)	79.097	-	-	-	-	-
Forward Propagation (Average)	2.553	2.872	4.840	9.231	12.827	19.713
Forward Propagation (Min)	2.213	2.564	4.003	5.478	8.301	14.733
Forward Propagation (Max)	2.693	3.092	6.436	12.566	15.324	21.815
Number of frame	31	31	31	31	31	31
Process per frame (second)	3.322	3.800	3.110	2.976	2.861	3.208

Table 5.2: Video Processing with YOLO Model

- Limitation / Problem - Limited Resource - CPU clock speed - RAM - Power resources

- CPU - ARM architecture limitation on floating-point [http://tinyurl.com/y85ykaqa] - When the number of threads is increasing, image processing task is not consistently processed by core. - Because the given task has to wait while core switching and doing another task (context switching) - Thus, the given task requires more time to be finished - JNI - Calling JNI is expensive [ref]

Model	MobileNet SSD					
	Sequential Computing	Parallel Computing				
		1 Thread	2 Threads	4 Threads	6 Threads	8 Threads
Total Process Time (second)	7.132	8.237	6.873	6.270	5.137	5.064
Garbage Collector (second)	-	-	-	-	-	-
Process Time without GC	-	-	-	-	-	-
Forward Propagation (Total)	7.019	-	-	-	-	-
Forward Propagation (Average)	0.226	0.235	0.401	0.738	0.900	1.133
Forward Propagation (Min)	0.218	0.212	0.353	0.406	0.428	0.466
Forward Propagation (Max)	0.243	0.320	0.456	1.477	3.057	2.582
Number of frame	31	31	31	31	31	31
Process per frame (second)	0.230	0.266	0.222	0.202	0.166	0.163

Table 5.3: Video Processing with MobileNet SSD Model

- Multithread Performance Analysis - I/O in thread - If there is I/O operation in thread or loop, it will cost a lot of overhead - Out of memory - If let each thread hold the large variable, it will cost memory overhead. - We have to free the variables after used. Otherwise, the x+1 th thread will allocate another xx MB. - Young generation - If there are lot of variables that are initialled in loop, there will be a lot young generation in the heap. So, when the number of young generations is reaching the threshold, GC will correct the young generation (freeing garbage in young generation heap) which affect the performance. - GC - Caused by Native [https://developer.android.com/studio/profile/memory-profiler] - Bin is GC - 8 GB (available only 3.8 GB) - CPU hits 100- CPU drops when GC is started
- Because threads are paused (Yellow) when GC is collecting

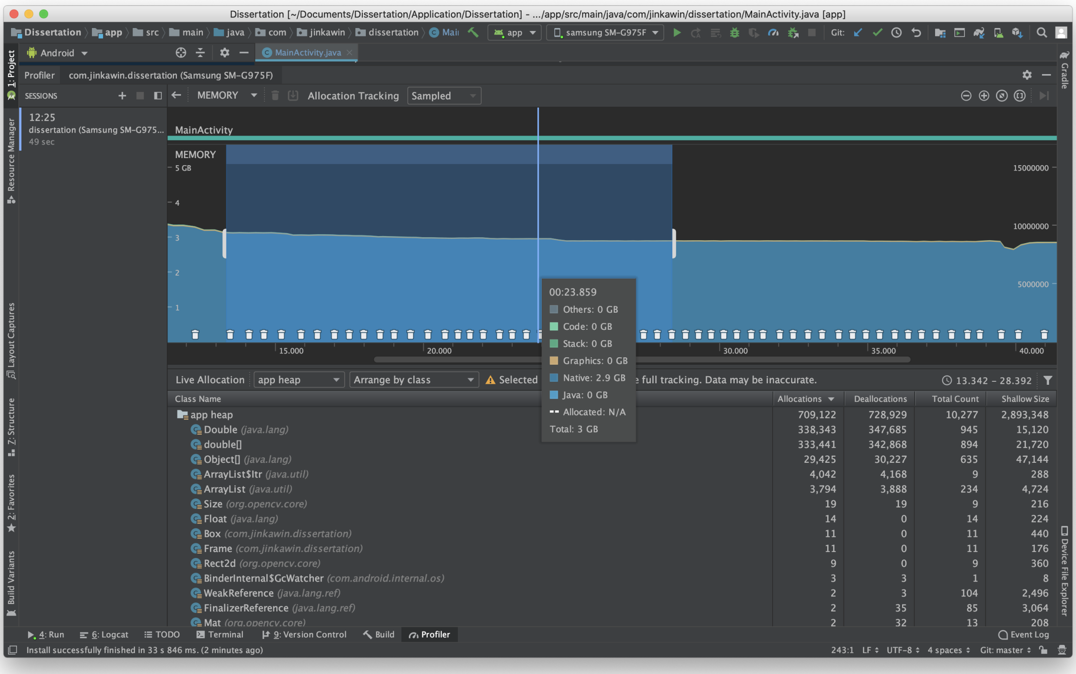

Figure 5.1: YOLO Model's Memory Usage

5.2 Usability Testing

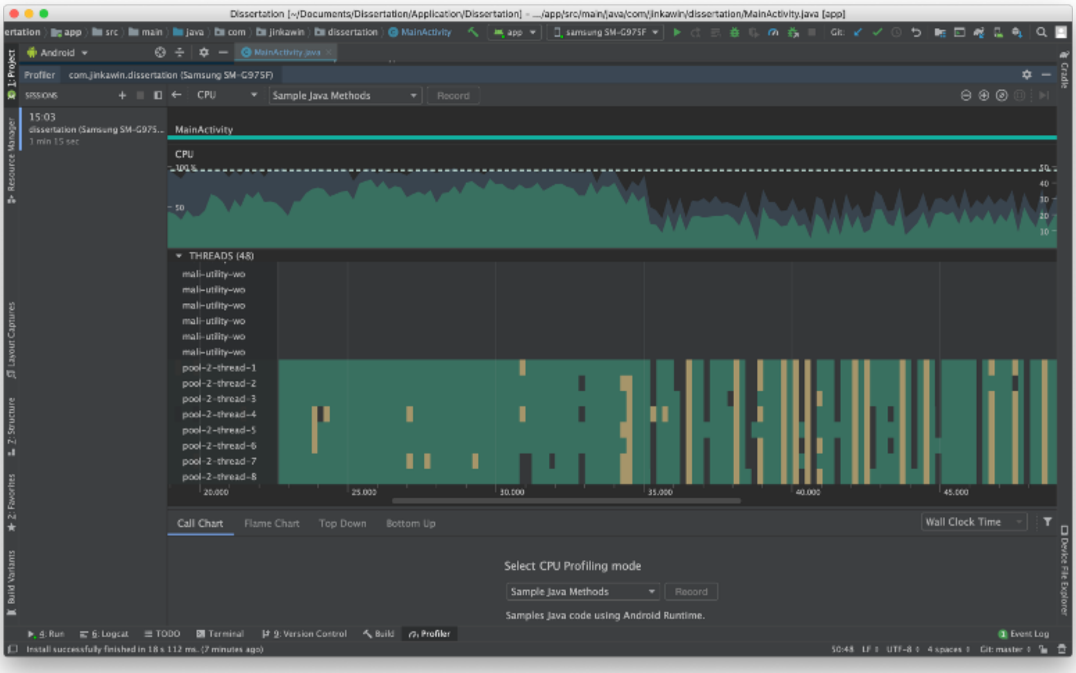

Figure 5.2: YOLO Model's CPU Usage

Chapter 6

Conclusion

6.1 Future Work

Appendix A

First appendix

A.1 Section of first appendix

Appendix B

Second appendix

Bibliography

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.