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

1.1 Motivation

Humanity have been faced several pandemics over hundreds years, and many lives are threatened. For example, there were the black death in 1346 and the flu pandemic in 1918, and millions people died during each pandemic [1], [5]. This year, humanity is confronting the other pandemic, which is named as coronavirus or COVID-19.

However, a threat from illness or the impact of the outbreak can be reduced by using technology and scientific discovery. For instance, social distancing has been recommended since there was Influenza A (H1N1) outbreaks in 2009 [4]. Social distancing is able to reduce the spread, and slow down and reduce the size of the epidemic peak [2], [3]. Consequently, an infection curve is flattened, and the number of deaths is reduced. Currently, scientific researchers are still working on this pendamic with the aim of reducing the infection rate. In addition, technology can be integrated with scientific theory to gain more advantages.

Due to competitive advantage and business competition, an ability of computable devices have been improved which is capable of executing very complex tasks. Mobile phone, which is one of a high competitive market, become a powerful device. People use a mobile phone for many purposes, such as entertainment, education or business. Likewise, in an application development, there are advantages of the mobile device. The first advantage is a performance. As aforementioned competition, hardware of mobile phone has been improved over years, including: CPU, GPU, and Memory. Thus, a capability of the mobile phone is sufficient for performing heavily calculation tasks Then, mobile phone is portability. Most of mobile phones provide necessary features, such as camera, sensor, and GPS. In addition, a mobile phone has a battery, which does not require a charger during being used.

For the advantages above, an mobile application can be used as a tool to help human determine social distancing.

1.2 Aim and Objectives

The aim of this project is determining distances between people, and maximising the performance of the application to achieve the highest frame rate (frame per second or FPS). To accomplish the aim, objectives of this project are specified by following:

- The application will determine objects from the given image and video by using Deep Neural Network (DNN).
- The application will determine distances between people in real-time from a camera.
- Input will be processed simultaneously by using parallel processing techinique.
- An advanced single-instruction multiple-data (SIMD), NEON instructions, will be implemented to improve the calculation performance.

1.3 Structure

The content of this dissertation is structured as follows:
1. Chapter 2 provides a background knowlegde of Android application, object detection, distance calculation, and parallel computing.
2. Chapter 3 shows an overview of the system, class diagram, and design.
3. Chapter 4 describes implementation details and technical development.
4. Chapter 5 analyses development's problems and the performance evaluation.
5. Chapter 6 concludes results of the project, limitation, and future works.

Chapter 2

Background

This chapter aim to explain technologies, tools, and specification of a device that are used in this project. Then, this chapter gives an analysis of related applications.

2.1 Android Application

- 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] - What is JNI - Why I need this

2.2 Human Detection

- People Detection - Using DNN - What is DNN - How it works - How it works in this project - Insert diagram of DNN (Flow of Image Processing) - blobFronImage - forward - ...

2.3 Distance Calculation

To measure a distance between 2 people, the reference point of people are used for calculation. The reference point is the coordination of each people, which is the centre of the detection frame. The calculation formula is based on Euclidean distance.

$$d = \sqrt{(a_0 - b_0)^2 + (D/c) \times (a_1 - b_1)^2}$$

$$c = \frac{a_1 + b_1}{2}$$

However, three-dimensional space are captured into two-dimensional image, so depth and perspective are concerned. Thus, a couple variables are added into the formula. The first variable is D, which is the diagonal of the image. The second variable is c, which is a calibration. These 2 variables will determine the depth of people in the image.

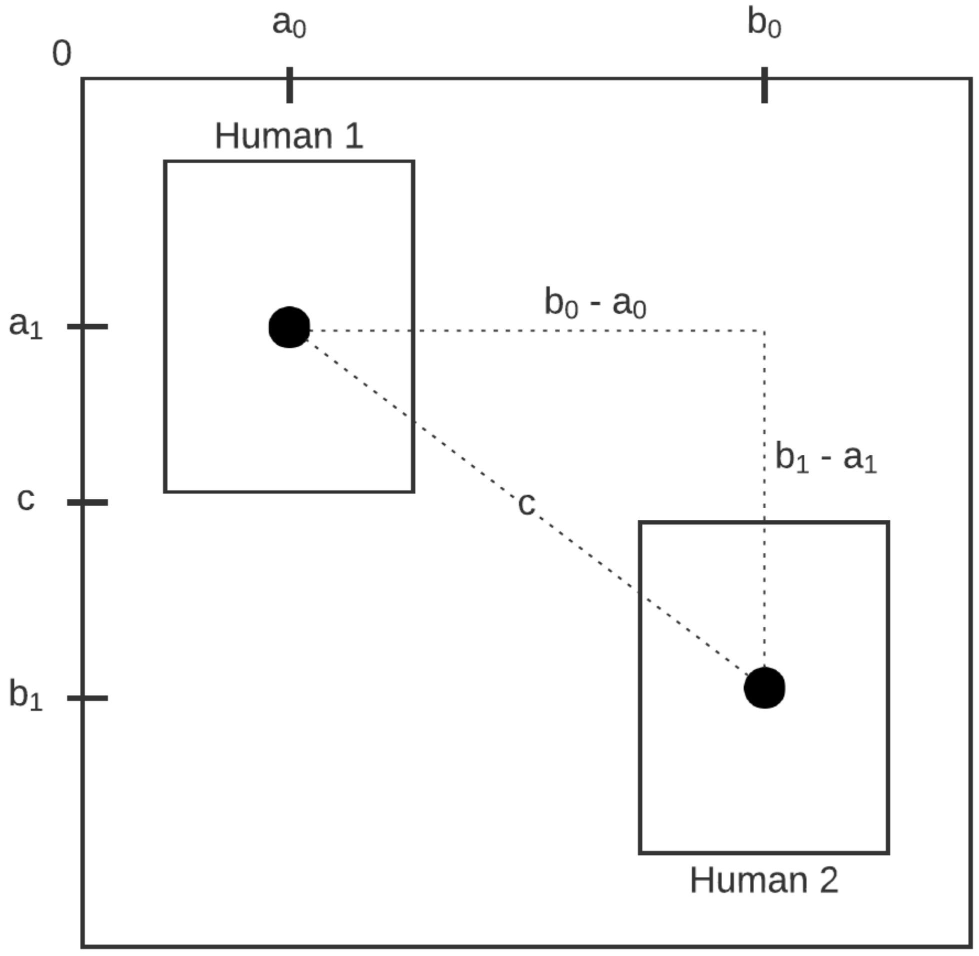


Figure 2.1: Distance Calculation

For example, according to the figure 2.2, if distance is calculated without calibration, distance between 2 couples will be the same. Naturally, the distance between Human1 and Human2 must be further than the distance between Human3 and Human4.

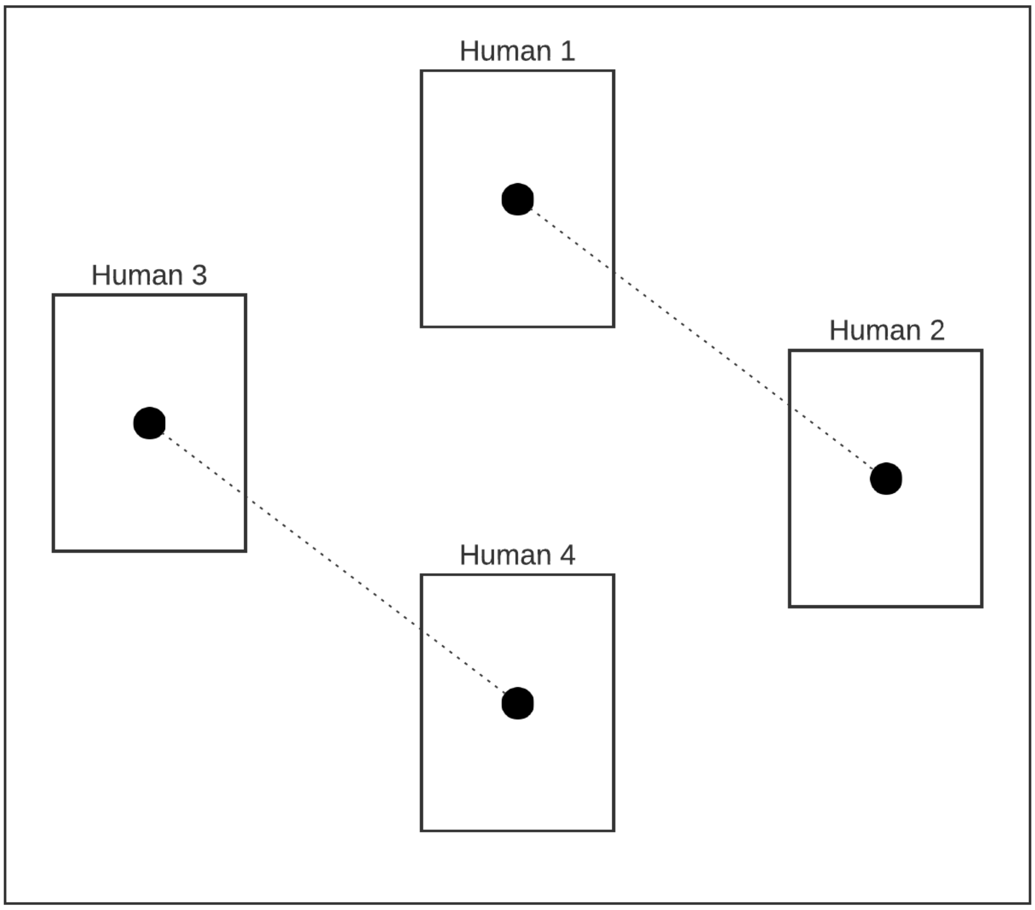


Figure 2.2: Distance Comparison

2.4 Parallel Computing

- What is parallel computing
- Multithreading
- Why I need it
- How it works
- new Thread() vs ThreadPool
- Insert general picture about Parallel computing
- Why it benefits for Android

2.5 Specification

This application is developed and tested on following specification:

Table 2.1: Specification

Device	Device	Samsung S10+
Operating System	Operating System	Android 10 (Q)
Processor	CPU	Samsung Exynos 9820
	Cores	8
	Architecture	2x ARM Cortex-A75 2.73GHz
		4x ARM Cortex-A55 1.95GHz
		2x Samsung Exynos M4 1.95 GHz
	GPU	Mali-G76
Memory	RAM	8 GB

2.6 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 MoSCoW Statement

3.1.1 Must have

- The application **must** be able to detect people in the given image or video.
- The application **must** be able to determind distancing between people in the given image or video.
- The application **must** save the processed image of video.
- The application **must** allow user to choose image or video from device's storage.

3.1.2 Should have

- The application **should** be able to stream video from camera.
- The application **should** be able to show detected people on camera.
- The application **should** support parallel computing.

3.1.3 Could have

- The application **could** choose computation options between sequencial or parallelism.
- The application **could** support NEON techonology.
- The application **could** be able to process the given tasks in background.

3.1.4 Won't have

- The application **won't** other objects which are not human.
- The application **won't** support GPU computation.

3.2 System Architecture

According to Figure 3.3

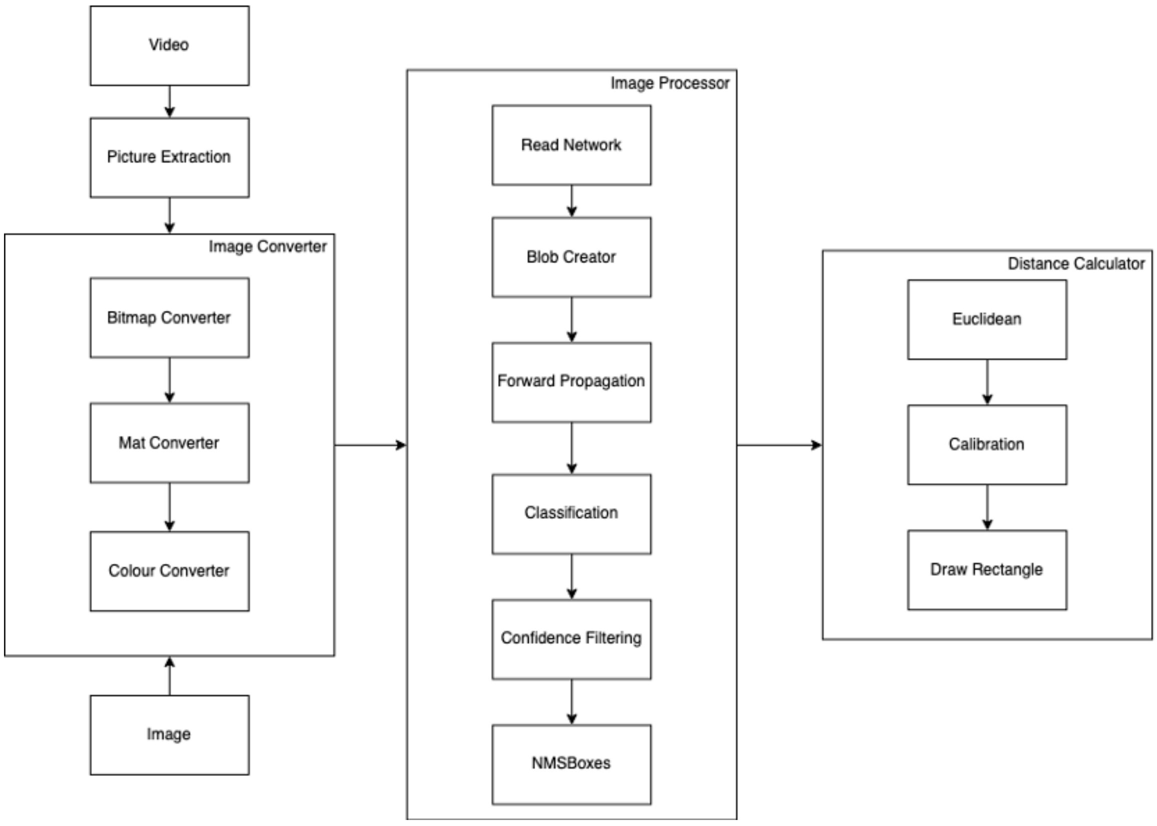


Figure 3.1: System Overview Diagram

3.3 Parallelism

3.4 Interface Design

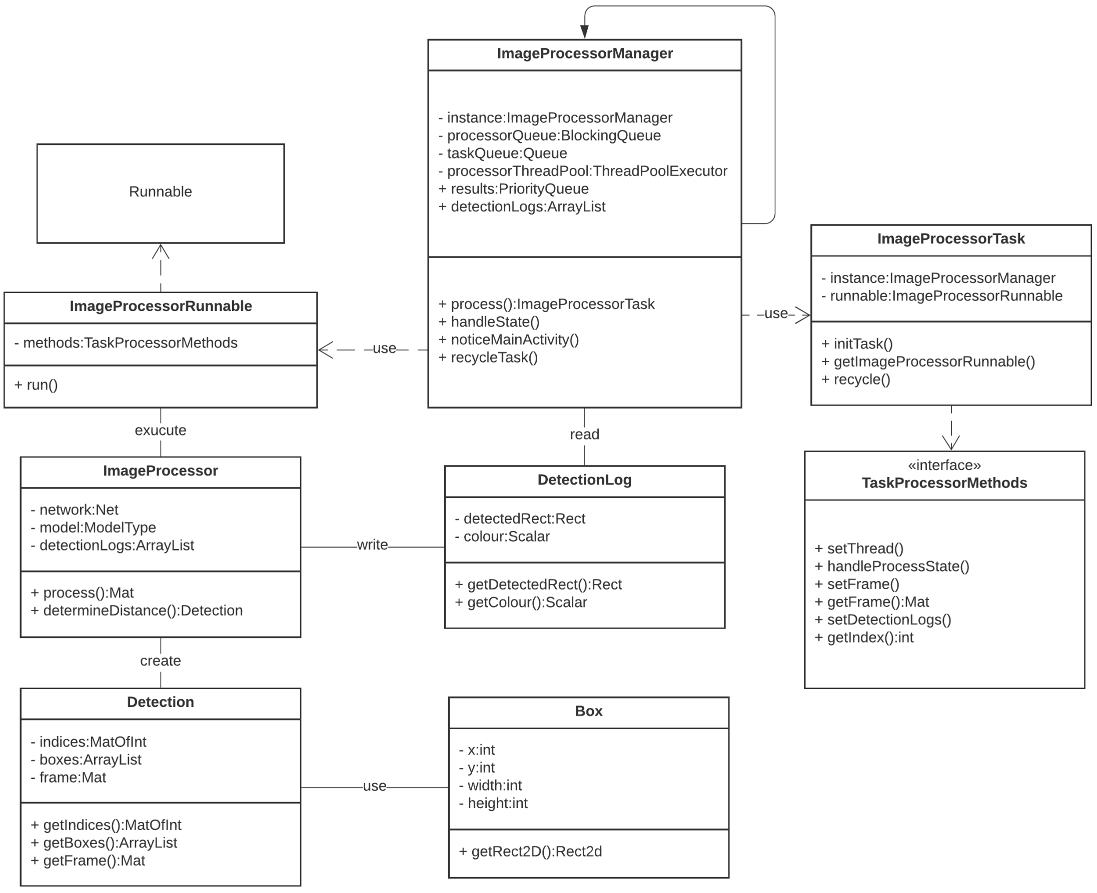


Figure 3.2: Detection UML Class

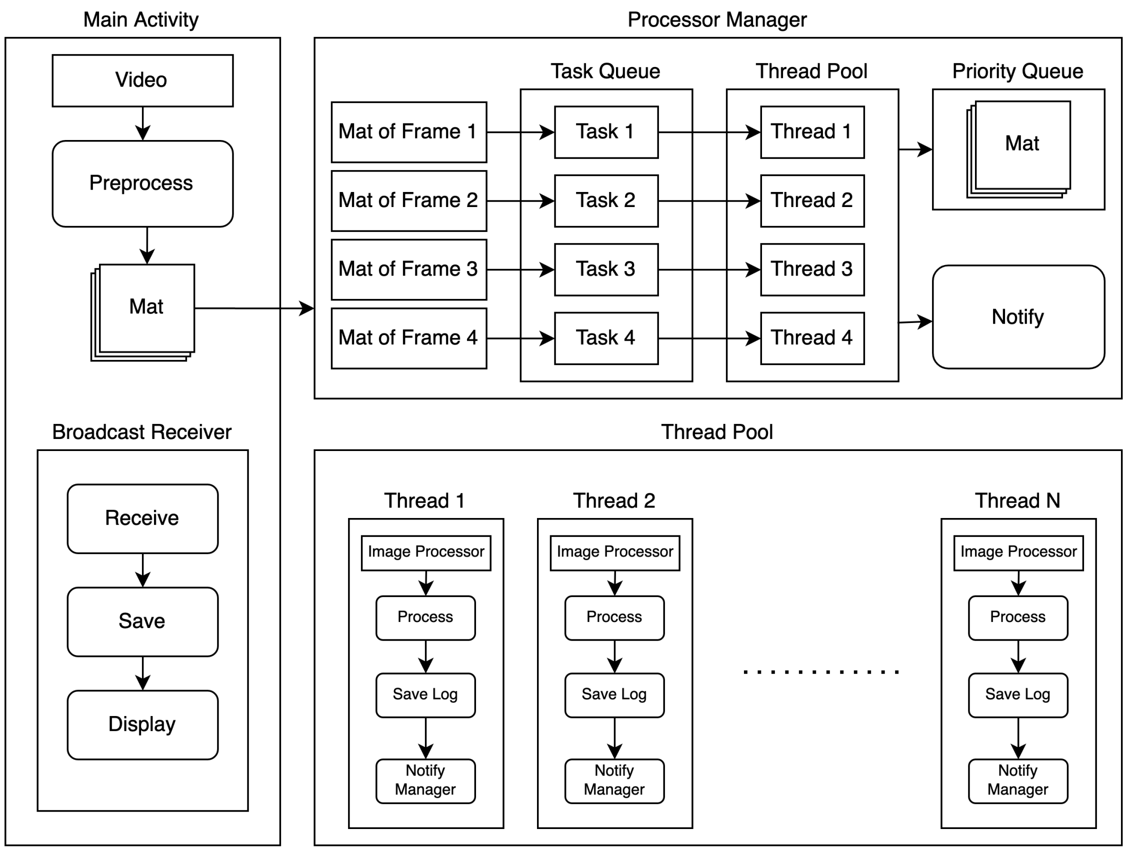


Figure 3.3: Parallel Computing Diagram

Chapter 4

Implement

In this chapter, the implementation details of each modules will be explained.

4.1 Android Application

This application is implemented on the Android Operating System, and the target Software Development Kit (SDK) version is set at level 29, namely Android Q. Implementation is divided into 3 layers. The first layer is an application layer, which is written in Java. This Layer mainly interacts with a user, checks permissions, and communicates with Java Native Interface (JNI). In addition, this layer also handle I/O implementation such as camera, file, and storage. The second layer is JNI, which is written in C and C++. JNI layer performs 2 main tasks. The fist task is being an intermediate connection between the application layer and a library layer. The second task is loading native shared libraries, which is compiled by a Native Development Kit (NDK). The last layer is the library layer, which is written in C++. This layer performs calculation tasks, including Deep Neural Network and distance calculation.

However, Deep Neural Network and distance calculation can be implemented in the application layer, but executing both tasks in the library layer gains a better performance. There are 2 reasons of increasing performance. The first reason is reducing JNI calling. Performing both tasks have to call JNI methods many times, and calling JNI methods are expensive and cost an overhead. Thus, implementing JNI manually reduces the number of JNI calls. The second reason is memory management. C++ is able to access values in the memory by using a pointer. Thus, values can be directly used without copying.

4.2 Human Detection

- There are 2 model which are used for doing forward propagation - YOLO3 Model - Mobilenet SSD Model - Confidence threashold level - YOLO Able to detect person with confidence threshold 0.5 - SSD Able to detect person with confidence threshold 0.3

- 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 Distance Calculation

- 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

4.4 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

This chapter presents an evaluation of the application, which is divided into 3 parts. The first part will show variables that must be controlled for the reliability and stability of the result. The second part is going to evaluate the application's performance, which are comparisons among models, programming languages, and technologies. The last section is going to show a usability testing of this application.

5.1 Controlled Variables

To ensure the result of the performance will not be varied by other factors, some variables must be controlled as following:

1. All testing cases will be run on Samsung Galaxy S10+.
2. All running background applications will be closed, and memory will be freed before testing.
3. To prevent CPU's speed is limited, a power management mode will be set to ”Optimized”.
4. Total number of frames in the testing video will be set to 31 frames.
5. A testing video resolution will be set to 540x480 pixels.

5.2 Performace Comparison

5.2.1 Model Comparison

In this section, the performance of 2 detection models will be compare, regardless of other tools and techniques. To evaluate the performance, each model will process on a single frame. There are 2 different resolutions were used as inputs for comparing the performance and understanding the variation of calculation time.

As can be seen from the Table 5.1, the processing time of YOLO model significantly increases when the size of the picture is greater. In contrast, MobileNet SSD is able to process the given picture

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

faster than YOLO model. The processing time of MobileNet SSD slightly increases when the size of the picture is increased.

5.2.2 Multithreading

In this section, the performance of both models will be evaluated with multithreading technique, and the evaluation will be divided into 3 parts: sequential computing, YOLO with multithreading, and MobileNet SSD with multithreading. As mentions previous section, in this evaluation, controlled variables are set. The number of frames in the testing video will be set to 31 frames.

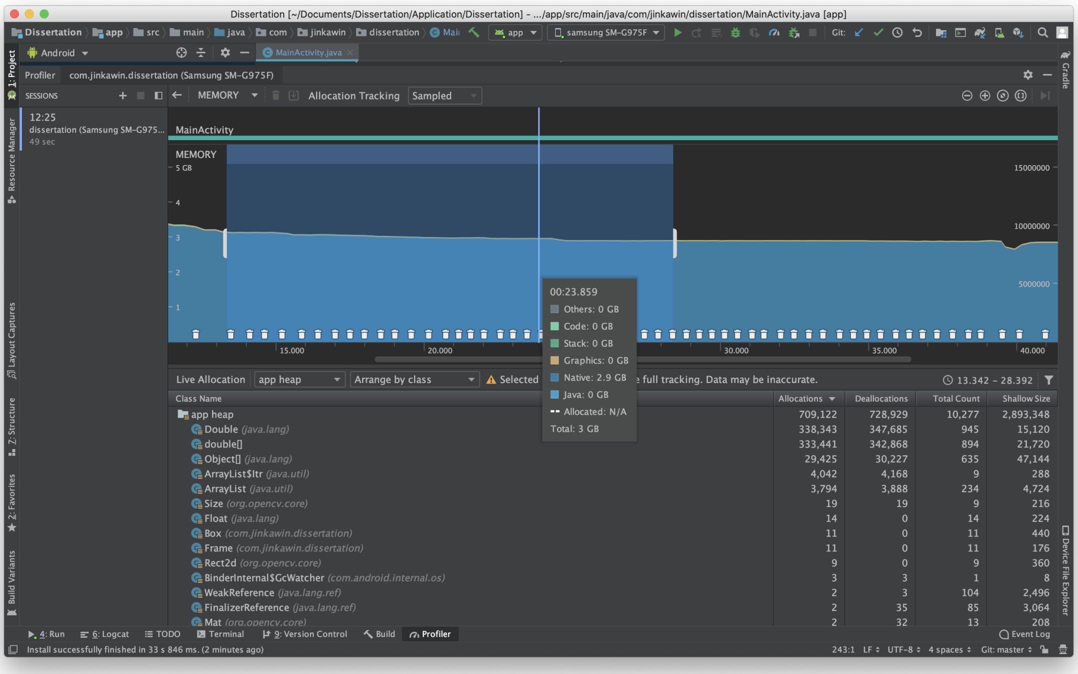
For the first part, an application will process the testing video sequentially, and measure the performance. This measurement will be compared to multithreading, and evaluate the improvement of the performance. As a result in Table 5.1 and 5.2, YOLO model took 102.972 seconds to processed 31 frames video, while MobileNet SSD model took 7.132 seconds. However, even if MobileNet SSD model can achieve better performance than YOLO model, the application is not able to process a video in real-time.

Model	YOLO				
	Sequential Computing	Parallel Computing			
		1 Thread	2 Threads	4 Threads	8 Threads
Total Process Time (second)	102.972	117.805	96.415	92.242	99.441
Garbage Collector (second)	-	0.102	0.280	2.024	11.333
Process Time without GC	-	117.703	96.136	90.218	88.108
Forward Propagation (Total)	79.097	-	-	-	-
Forward Propagation (Average)	2.553	2.872	4.840	9.231	19.713
Forward Propagation (Min)	2.213	2.564	4.003	5.478	14.733
Forward Propagation (Max)	2.693	3.092	6.436	12.566	21.815
Number of frame	31	31	31	31	31
Process per frame (second)	3.322	3.800	3.110	2.976	3.208
Improvement			18.16%	21.70%	15.59%

Table 5.2: Video Processing with YOLO Model with official build

Then, a multithreading technique is implemented to increase performance and achieve real-time processing. To evaluate the improvement of multithreading, a number of processors will be doubled as follows: 1, 2, 4, 8, and 16. In the testing device, there are 8 physical cores, so it can effectively process up to 8 threads.

The overall performace of YOLO model with multithreading is slightly improved. There are 2 problems of YOLO model with multithreading as shown in Table 5.2. The first problem is the improvement of the performance. The performance of 2 threads is improved only 18.16 percent, and it reach the best performance at 21.7 percent by using 4 threads. However, the performance of 8 threads is worse than 2 threads, which is caused by Garbage Collection. The second problem is Garbage Collection (GC). The application was frozen while GC is collecting garbage. GC is



Selected ... full tracking. Data may be inaccurate.

Class Name	Allocations	Deallocations	Total Count	Shallow Size
app heap	709,122	728,929	10,277	2,893,348
Double *(java.lang)*	338,343	347,685	945	15,120
double[]	333,441	342,868	894	21,720
Object[] *(java.lang)*	29,425	30,227	635	47,144
ArrayList$Itr *(java.util)*	4,042	4,168	9	288
ArrayList *(java.util)*	3,794	3,888	234	4,724
Size *(org.opencv.core)*	19	19	9	216
Float *(java.lang)*	14	0	14	224
Box *(com.jinkawin.dissertation)*	11	0	11	440
Frame *(com.jinkawin.dissertation)*	11	0	11	176
Rect2d *(org.opencv.core)*	9	0	9	360
BinderInternal$GcWatcher *(com.android.internal.os)*	3	3	1	8
WeakReference *(java.lang.ref)*	2	3	104	2,496
FinalizerReference *(java.lang.ref)*	2	35	85	3,064
Mat *(org.opencv.core)*	2	32	13	208

Install successfully finished in 33 s 846 ms. (2 minutes ago)

Figure 5.1: Memory Usage of YOLO Model with 8 threads

not collecting only short-lived objects but long-lived objects as well, and GC is more often collect garbage when the number of threads is increased. As shown in Figure 5.1, memory was allocated by double and array of double, and GC was freeing these allocation 10 times within 5 seconds. Consequently, CPU usage is dropped when GC is working. This problem can be seen in the Figure 5.2. The progress status will be green when a thread is working. It will be yellow when it is interrupted by GC, and it will be gray when a thread has no activity.

Model	SSD, OpenCV 3.4.0 Official Build				
	Sequential Computing	Multithreading			
		1 Thread	2 Threads	4 Threads	8 Threads
Total Process Time (second)	7.132	8.237	6.873	6.270	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	1.133
Forward Propagation (Min)	0.218	0.212	0.353	0.406	0.466
Forward Propagation (Max)	0.243	0.320	0.456	1.477	2.582
Number of frame	31	31	31	31	31
Process per frame (second)	0.230	0.266	0.222	0.202	0.163
Improvement			16.56%	23.88%	38.52%

Table 5.3: Video Processing with MobileNet SSD Model with official

For the MobileNet SSD model, Table 5.3 compares the computation time among 1, 2, 4, and 8 threads. The calculation time of 2 threads is improved only 16.56 percent, and 8 threads give the best performance with 38.53 percent. The improvement time of 8 threads by using MobileNet SSD model is not reduces likes YOLO model, but speedup time of both models still lower than an ideal time.

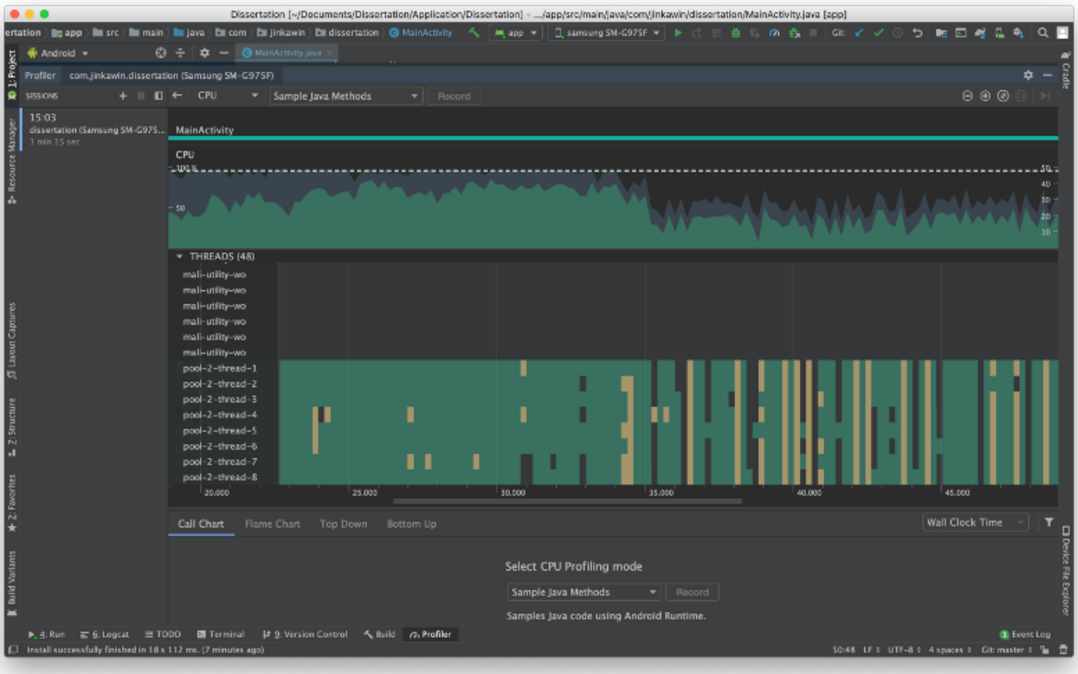


Figure 5.2: CPU Usage of YOLO Model with 8 threads

The performance of multithreading is slightly improved, and it is not able to achieve theoretical speedup of Amdahl's law by using multithreading.

5.2.3 NEON Comparison

Model	MobileNet SSD without NEON				
	Sequential Computing	Parallel Computing			
		1 Thread	2 Threads	4 Threads	8 Threads
Total Process Time (second)	17.308	19.030	15.172	11.797	10.624
Garbage Collector (second)	-	-	-	-	-
Process Time without GC	-	-	-	-	-
Forward Propagation (Total)	17.193	17.848	-	-	-
Forward Propagation (Average)	0.555	0.576	0.926	1.416	2.462
Forward Propagation (Min)	0.519	0.545	0.582	0.756	1.310
Forward Propagation (Max)	0.586	0.654	1.412	2.593	5.308
Number of frame	31	31	31	31	31
Process per frame (second)	0.558	0.614	0.489	0.381	0.343
Improvement			20.27%	38.01%	44.17%

Table 5.4: Video Processing with MobileNet SSD Model without NEON

- MobileNet is faster, but less accuracy.

Model	MobileNet SSD with NEON				
	Sequential Computing	Multithreading			
		1 Thread	2 Threads	4 Threads	8 Threads
Total Process Time (second)	4.006	6.079	4.208	3.127	2.890
Garbage Collector (second)	-	-	-	-	-
Process Time without GC	-	-	-	-	-
Forward Propagation (Total)	3.927	4.950	-	-	-
Forward Propagation (Average)	0.126	0.159	0.225	0.339	0.645
Forward Propagation (Min)	0.117	0.128	0.131	0.190	0.266
Forward Propagation (Max)	0.135	0.250	0.295	0.596	1.798
Number of frame	31	31	31	31	31
Process per frame (second)	0.129	0.196	0.136	0.101	0.093
Improvement			30.78%	48.56%	52.46%

Table 5.5: Video Processing with MobileNet SSD Model with NEON

5.3 Usability Testing

- FPS on video

Chapter 6

Conclusion

6.1 Limitation

- 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 - There are stages of CPU's clock frequency - JNI - Calling JNI is expensive [ref]

- 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

- Thread Pool Problem - Thread Pool only improve when there are large number of asynchronous tasks. [https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ThreadPoolExecutor.html]

6.2 Future Work

Appendix A

First appendix

A.1 Section of first appendix

Appendix B

Second appendix

Bibliography

[1] Ole J. Benedictow. *The Black Death, 1346-1353: the complete history*. Boydell Press, Woodbridge, Suffolk, 2004.

[2] Min W. Fong, Huizhi Gao, Jessica Y. Wong, Jingyi Xiao, Eunice Y. C. Shiu, Sukhyun Ryu, and Benjamin J. Cowling. Nonpharmaceutical measures for pandemic influenza in nonhealthcare settings—social distancing measures. *Emerging infectious diseases*, 26(5):976–984, 2020.

[3] Centers for Disease Control and Prevention. Social distancing, 2020.

[4] Jennifer A. Horney, Zack Moore, Meredith Davis, and Pia D. M. MacDonald. Intent to receive pandemic influenza a (h1n1) vaccine, compliance with social distancing and sources of information in nc, 2009. *PloS one*, 5(6):e11226, 2010.

[5] Ann H. Reid, Raina M. Lourens, Ruixue Wang, Guozhong Jin, Jeffery K. Taubenberger, and Thomas G. Fanning. Molecular virology was the 1918 pandemic caused by a bird flu? was the 1918 flu avian in origin? (reply). *Nature*, 440(7088):E9–E10, 2006.