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

Acknowledgements

acknowledgements go here

Contents

1 Introduction **5**

1.1 Motivation . . . 5

1.2 Objectives . . . 5

1.3 Structure . . . 6

2 Background **7**

2.1 Detection Algorithm . . . 7

2.1.1 Deep Neural Network . . . 7

2.2 Mobile Technology . . . 7

2.3 Parallel Computing . . . 8

2.4 Specification . . . 8

2.5 Existing Applications . . . 8

3 Design **9**

3.1 MoSCoW Statement . . . 9

3.1.1 Must have . . . 9

3.1.2 Should have . . . 9

3.1.3 Could have . . . 9

3.1.4 Won't have . . . 9

3.2 System Architecture . . . 10

3.3 Parallelism . . . 10

3.4 Interface Design . . . 10

4 Implement **12**

4.1 Android Application 12

4.2 Deep Neutral Network 12

4.3 Deep Neutral Network 13

4.4 Parallelisation 13

4.4.1 Multitheading with CPU 13

5 Testing and Evaluation **14**

5.1 Performace Evaluation 14

5.2 Usability Testing 15

6 Conclusion **17**

6.1 Future Work 17

A First appendix **18**

A.1 Section of first appendix 18

B Second appendix **19**

Chapter 1

Introduction

1.1 Motivation

There are pandemics over hundreds years, including flu pandemic in 1918 and the black death in 1346 [5], [1]. There were millions people died during each pandemic. This year, humanity is facing the other pandemic, which is named as coronavirus or COVID-19. However, COVID-19 can be well-managed with scientific discovery and technology. For instance, social distancing has been recommended since there was Influenza A (H1N1) pandemic in 2009 [4]. It 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. Yet, scientific researchers still work continuously on this outbreak. In addition, technology can be integrated with scientific theory to gain advantages.

- 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

The content of this dissertation is structured as follows: 1. Chapter 2 provides a background knowlegde of deep neural network, parallel computing, and mobile technology.
2. Chapter 3 shows an overview of the system, data flow, and design
3. Chapter 4 describes implementation
4. Chapter 5 provides results and analysis
5. Chapter 6 concludes xxx, 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 Detection Algorithm

2.1.1 Deep Neural Network

- 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 - ... - Distance Calculation - Explain about formula

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

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

- C value (Calibation) -¿ perspective - Image diagonal line (550 -¿ (480x270)) - Euclidean Distance - Insert picture for explination - Lib - Models

2.2 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] - What is JNI - Why I need this

2.3 Parallel Computing

- What is parallel computing - Why I need it - How it works - new Thread() vs ThreadPool - 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 MoSCoW Statement

3.1.1 Must have

- The application **must** be able to detect person 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 stram video from camera.
- The application **should** be able to show detected person 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 3.2

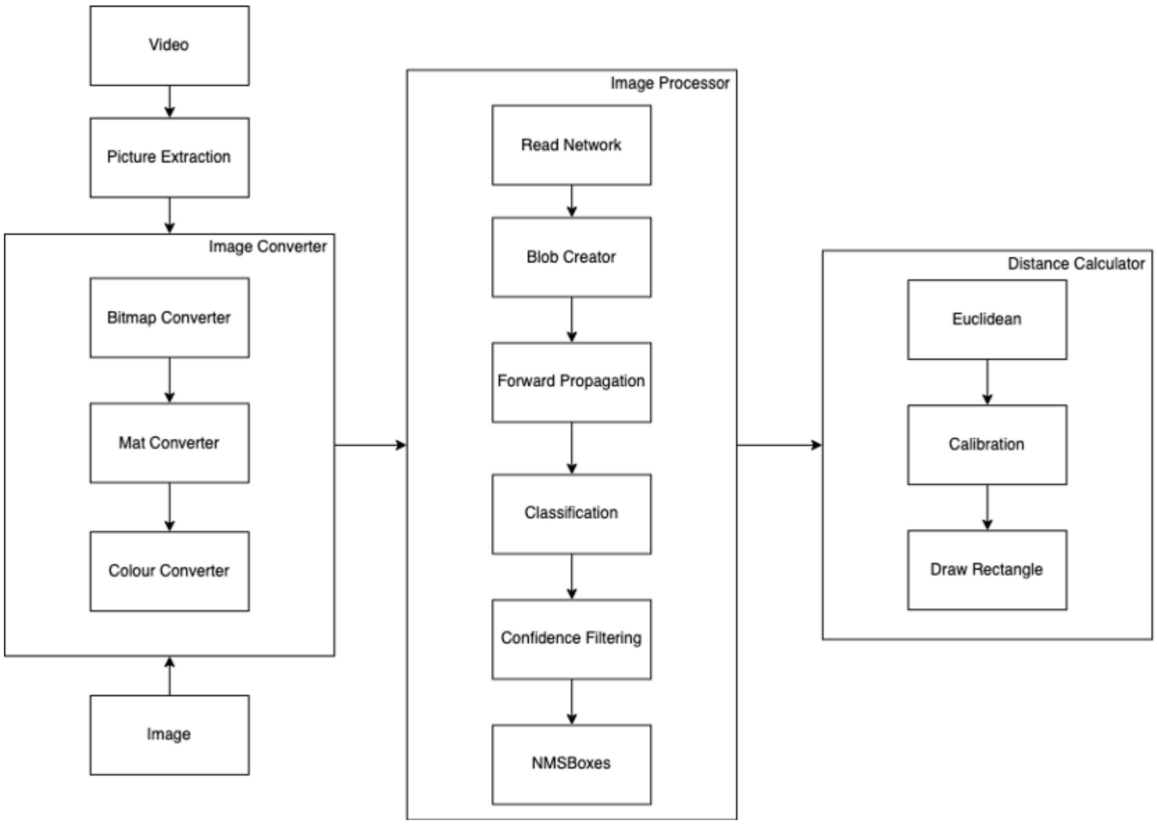


Figure 3.1: System Overview Diagram

3.3 Parallelism

3.4 Interface Design

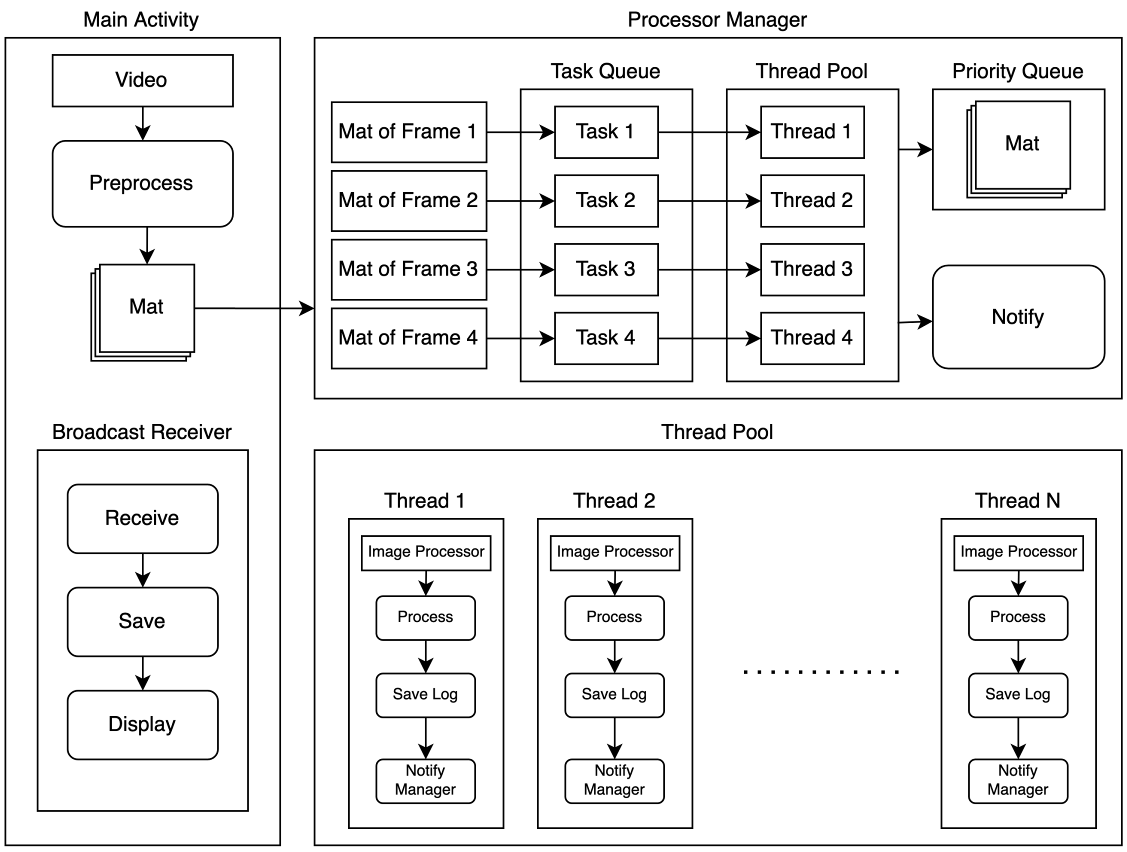


Figure 3.2: Parallel Computing Diagram

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 gaining the better 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 memeory by using a pointer. Thus, values can be directly used without copying.

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)

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

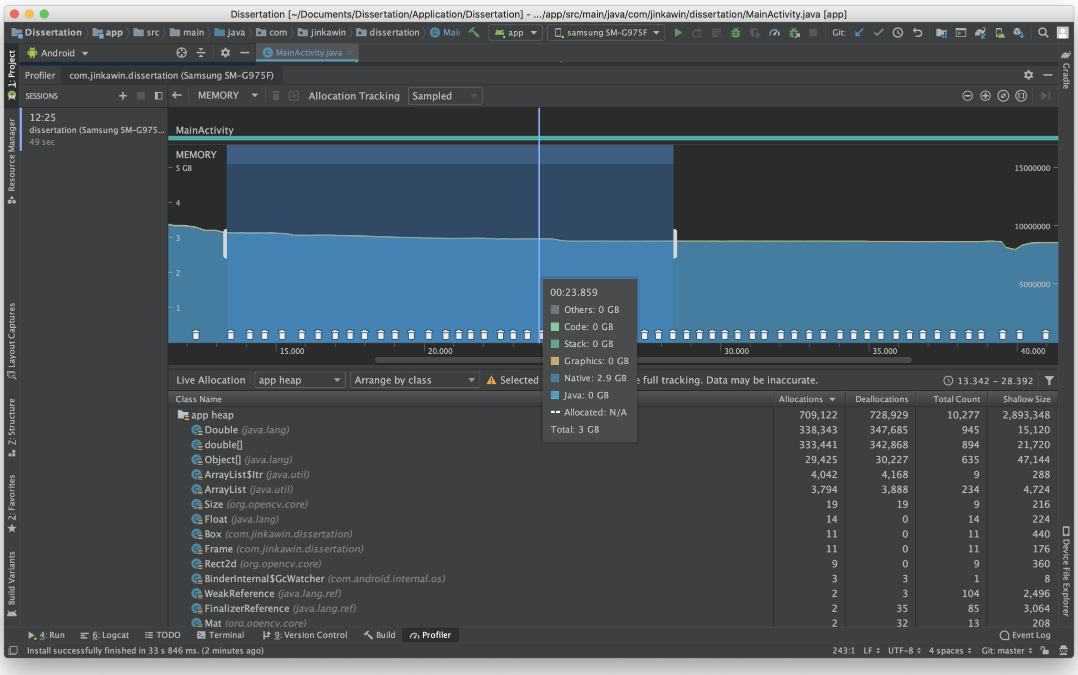
Table 5.3: Video Processing with MobileNet SSD Model

- 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]

5.2 Usability Testing

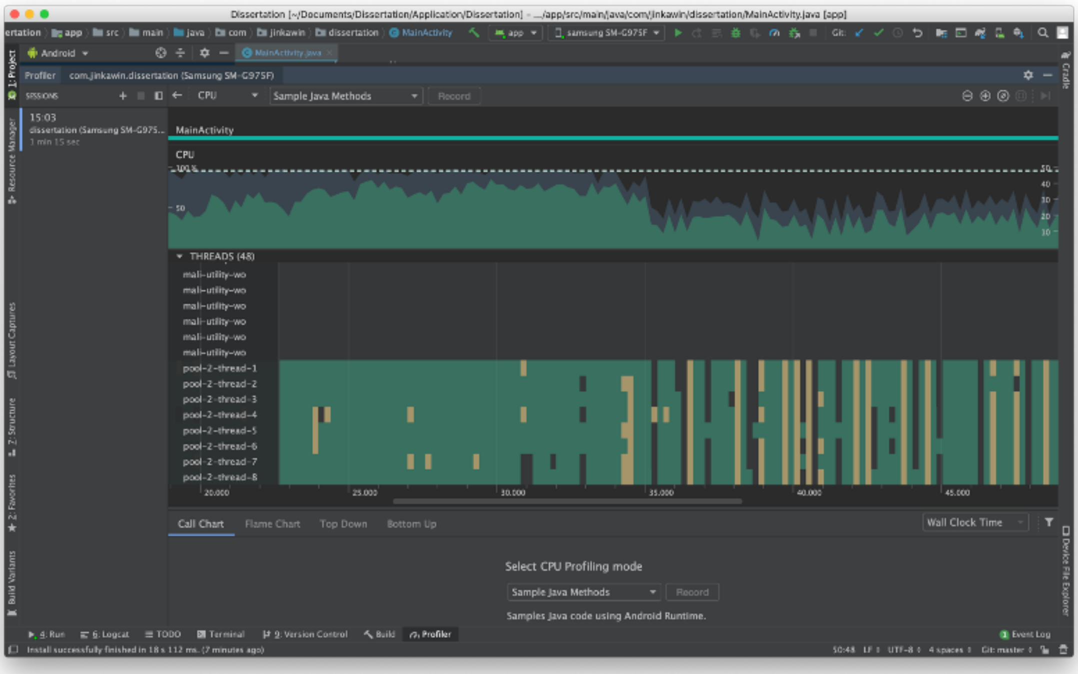

00:23.859
Others: 0 GB
Code: 0 GB
Stack: 0 GB
Graphics: 0 GB
Native: 2.9 GB
Java: 0 GB
Allocated: N/A
Total: 3 GB

Live Allocation | app heap | Arrange by class | Selected … full tracking. Data may be inaccurate. | 13.342 – 28.392

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: YOLO Model's Memory Usage



Select CPU Profiling mode

Sample Java Methods | Record

Samples Java code using Android Runtime.

```
Install successfully finished in 18 s 112 ms. (7 minutes ago)
```

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] 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.