

**VISUAL CROWD COUNTING SYSTEM USING DEEP LEARNING**  
**BY**  
**MOHD WAFI NAZRUL ADAM BIN MOHD RIDHWAN OXLEY ADAM**

**A REPORT**  
**SUBMITTED TO**  
Universiti Tunku Abdul Rahman  
in partial fulfillment of the requirements  
for the degree of  
**BACHELOR OF INFORMATION TECHNOLOGY (HONOURS)**  
**COMPUTER ENGINEERING**  
Faculty of Information and Communication Technology  
(Kampar Campus)

**MAY 2021**

## REPORT STATUS DECLARATION FORM

Title: Visual Crowd Counting System Using Deep Learning

Academic Session: May 2021

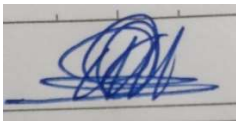
I Mohd Wafi Nazrul Adam Bin Mohd Ridhwan Oxley Adam

(CAPITAL LETTER)

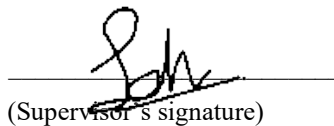
declare that I allow this Final Year Project Report to be kept in  
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

**Address:**

No.4, Jalan 17/54,

Taman Tan Sri Lee Yan Lian

Petaling Jaya, Selangor

Teoh Shen Khang

Supervisor's name

Date: 29<sup>th</sup> of August 2021

Date: 3 September 2021

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**  
**UNIVERSITI TUNKU ABDUL RAHMAN**

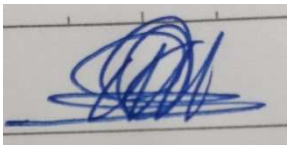
Date: 29<sup>th</sup> of August 2021

**SUBMISSION OF FINAL YEAR PROJECT**

It is hereby certified that **Mohd Wafi Nazrul Adam Bin Mohd Ridhwan Oxley Adam** (ID No: **17ACB05930**) has completed this final year project/ dissertation/ thesis\* entitled “**Visual Crowd Counting System Using Deep Learning**” under the supervision of Mr. Teoh Shen Khang (Supervisor) from the Department of Computer and Communication Technology, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project / dissertation/ thesis\* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,



(Mohd Wafi Nazrul Adam Bin Mohd Ridhwan Oxley Adam)

\*Delete whichever not applicable

**VISUAL CROWD COUNTING SYSTEM USING DEEP LEARNING**  
**BY**  
**MOHD WAFI NAZRUL ADAM BIN MOHD RIDHWAN OXLEY ADAM**

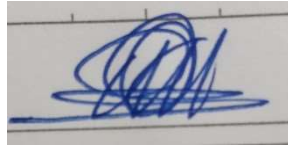
**A REPORT**  
**SUBMITTED TO**  
Universiti Tunku Abdul Rahman  
in partial fulfillment of the requirements  
for the degree of  
**BACHELOR OF INFORMATION TECHNOLOGY (HONOURS)**  
**COMPUTER ENGINEERING**  
Faculty of Information and Communication Technology  
(Kampar Campus)

**MAY 2021**

## DECLARATION OF ORIGINALITY

I declare that this report entitled “**VISUSAL CROWD COUNTING SYSTEM USING DEEP LEARNING**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :



Name :

Mohd Wafi Nazrul Adam Bin Mohd Ridhwan Oxley Adam

Date :

29<sup>th</sup> of August 2021

## **ACKNOWLEDGEMENTS**

I would like to express my gratitude and appreciation to my supervisor, Mr. Teoh Shen Khang and my moderator, Mr. Leong Chun Farn who have blessed me with the knowledge in embedded systems and digital circuits during my years as a Computer Engineering student in Faculty of Information and Computer Technology, Universiti Tunku Abdul Rahman. The skills I have honed under their guidance has helped me excel in the completion of this project.

Other than that, I would also like to thank my fellow course mates, Pravinden A/L Gunarajan, Yong Yi Jie and Tan E-Chian for the immense support and assistance throughout our time in Universiti Tunku Abdul Rahman.

## ABSTRACT

This project is about developing a visual crowd counting system using deep learning. The entirety of this project will only be using Python for both the back-end and the front-end development. The goal of this project is to develop a working system that could take in images and estimate the number of crowds in those images as well as display its estimated density map and a graph of predicted count against its ground truth as well as its accuracy in Mean Absolute Error (MAE) and Mean Squared Error (MSE). The back-end will be using a neural network model based on the Single-Image Crowd Counting via Multi-Column Convolutional Neural Network (Zhang, et al., 2016) and is developed through the PyTorch framework, an open-source machine learning library. The model will be trained using the Mall Dataset and the Adam optimization algorithm. The trained model has an accuracy of 2.45 in MAE and 9.72 in MSE when tested using the Test portion of the dataset. The front-end is developed from scratch using the PyQT5 toolkit and QtDesigner.

## TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>REPORT STATUS DECLARATION FORM</b>	<b>ii</b>
<b>FYP THESIS SUBMISSION FORM</b>	<b>iii</b>
<b>DECLARATION OF ORIGINALITY</b>	<b>v</b>
<b>ACKNOWLEDGEMENTS</b>	<b>vi</b>
<b>ABSTRACT</b>	<b>vii</b>
<b>TABLE OF CONTENTS</b>	<b>viii – ix</b>
<b>LIST OF FIGURES</b>	<b>x – xi</b>
<b>LIST OF TABLES</b>	<b>xii</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xiii</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1 – 8</b>
1.1 Problem Statement and Motivation	1
1.2 Objectives	1
1.3 Proposed Approach	2
1.4 Highlight of What Have Been Achieved	5
1.5 Report Organization	8
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>9 – 13</b>
1.1 Detection based methods	9
1.2 Regression based methods	10
1.3 Machine Learning methods	12
1.4 Summary	13
<b>CHAPTER 3 SYSTEM DESIGN</b>	<b>14 – 27</b>
<b>CHAPTER 4 SYSTEM IMPLEMENTATION</b>	<b>28 – 34</b>
1.1 Methodology	28
1.2 System Performance Definition	29
1.3 Verification Plan	30
1.4 System Overview	30
1.5 Timeline	34



<b>CHAPTER 5 SYSTEM EVALUATION AND DISCUSSION</b>	<b>35 – 37</b>
1.1 Test Case	35
1.2 Discussion	37
<b>CHAPTER 6 CONCLUSION</b>	<b>38</b>
1.1 Project Review	38
1.2 Future Work	38
<b>REFERENCES</b>	<b>39</b>
<b>WEEKLY LOG</b>	<b>40 – 45</b>
<b>POSTER</b>	<b>46</b>
<b>PLAGIARISM CHECK RESULT</b>	<b>47 – 48</b>
<b>CHECKLIST</b>	<b>49</b>

## LIST OF FIGURES

<b>Fig. No.</b>	<b>Title</b>	<b>Page</b>
Figure 1-3-1	Gaussian Density Map Formation	2
Figure 1-3-2	How train.py utilizes dataloader.py and model2.py together	3
Figure 1-3-3	Sketch of the GUI	3
Figure 1-3-4	How the system files work with each other	4
Figure 1-4-1	Graph between Prediction and Ground Truth	5
Figure 1-4-2	MAE and MSE of the trained neural network	5
Figure 1-4-3	System displaying estimated density map	6
Figure 1-4-4	System displaying estimated crowd count and graph analysis	6
Figure 1-4-5	System displaying estimated crowd count without ground truth	7
Figure 1-4-6	System displaying estimated crowd count of video feed	7
Figure 2-1	Sample images from their human detection database	9
Figure 2-2	Omega salient feature (left), sample images used in training set	10
Figure 2-3	Outline of Process	11
Figure 3-1	Gaussian Density Map Formation	14
Figure 3-2	Code snippet of the neural network model	14
Figure 3-3	How train.py utilizes dataloader.py and model2.py together	15
Figure 3-4	Code snippet of newui.py	16
Figure 3-5	Sketch of the GUI	16
Figure 3-6	Command line to convert .ui to .py	16
Figure 3-7	Image snippet of the system	17
Figure 3-8	Code snippet of newui.py regarding its connection to method	17
Figure 3-9	Code snippet of newui.py regarding the aforementioned method	17
Figure 3-10	Code snippet of the select image button	17
Figure 3-11	Code snippet of the method connected to select image button	18
Figure 3-12	Code snippet of the include ground truth checkbox	18
Figure 3-13	Code snippet of the select .csv file	19
Figure 3-14	Code snippet of the include ground truth checkbox	19
Figure 3-15	Code snippet of the method connected to the turn on/off video	19
Figure 3-16	Camera and QTimer instantiation	20

## LIST OF FIGURES

<b>Fig. No.</b>	<b>Title</b>	<b>Page</b>
Figure 3-17	Camera class	20
Figure 3-18	Methods responsible for the live video feed funct.	21
Figure 3-19	Code snippet of the show density map button	21
Figure 3-20	Code snippet of the method connected to the button	21
Figure 3-21	Code snippet of the predict button	22
Figure 3-22	Code snippet of the method connected to the predict but.	22
Figure 3-23	Code snippet of the required libraries and imports	23
Figure 3-24	Predict method that instantiates the neural network	23
Figure 3-25	Method of predictV that predicts the live video feed count	24
Figure 3-26	estimate_density_map method code snippet	25
Figure 3-27	Graph method that displays the graph figure, MAE/MSE	26
Figure 3-28	Diagram of how the system works	27
Figure 4-1	SDLC Process Flowchart	28
Figure 4-2	Mean Absolute Error Equation	29
Figure 4-3	Mean Squared Error Equation	30
Figure 4-4	Structure of the proposed mutli-column convolutional Neural network	31
Figure 4-5	Example of the ground truth density map	31
Figure 4-6	Image of the system	33
Figure 4-7	Gantt Chart of the timeline	34

## LIST OF TABLES

<b>Table No.</b>	<b>Title</b>	<b>Page</b>
Table 5.1	Test Plan of the System	35

## LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
FAIR	Facebook's AI Research Lab
GMM	Gaussian Mixture Model
GP	Gaussian Process
GPU	Graphics Processing Unit
GUI	Graphical User Interface
LBP	Local Binary Pattern
MAE	Mean Absolute Error
MSE	Mean Squared Error
SDLC	Software Design Life Cycle
SIFT	Scale-invariant feature transform

## **CHAPTER 1 : INTRODUCTION**

### **1.1 Problem Statement and Motivation**

During the development and testing of a neural network model, the developer has to create a separate file to test its accuracy which takes a lot of time during the development. Furthermore, there is rarely a graphical user interface (GUI) for it too, making the developer harder to infer the output data from the tests.

As a result, this project could help expedite the development by creating a ready-to-use system that could take in the parameter file, ground truth comma separated value (CSV) file and image files to output it's estimated density map, estimated count and a graph analysis of its accuracy as well as the accuracy in Mean Absolute Error (MAE) and Mean Squared Error (MSE).

Not only that, once the development is complete, the system can also be used as a standalone system to calculate the crowd count.

The motivation for this project is to enhance my programming skills in Python and also catch a glimpse of the development of deep learning specifically the convolutional neural network amidst the expansion of Industrial Revolution 4.0. This project will also serve as proof to would-be job providers that I have the ability and motivation to learn something new.

### **1.2 Objectives**

The objectives for this project are to train a neural network model to estimate the number of people within crowd using the PyTorch framework. The neural network that will be used is called MCNN developed by (Zhang, et al., 2016) but modified to implement batch normalization within the neural network as well. The dataset that will be used to train the said model is the Mall Dataset, an annotated dataset that has the coordinates of the heads of the people in the images and are marked and stored in a .mat file. The image dataset is then applied a gaussian kernel filter to create a ground truth density map where it will be fed along with its corresponding images to the neural network model for training.

After training is completed, the model would then create the image's density map and estimate the number of crowds from there. The accuracy of the model would be calculated using

## Chapter 1: Introduction

the Mean Absolute Error and Mean Squared Error as it is the standard for most of the crowd counting algorithms out there.

After that, a GUI will be implemented to communicate with the neural network to help facilitate the input of images to be estimated, parameter files and optionally the ground truth file for testing. The front-end development will be done using PyQt5 toolkit which will be programmed in Python as well.

### Objectives

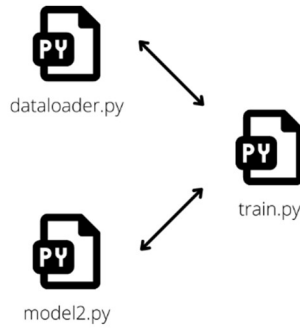
- Develop a slightly modified MCNN in PyTorch
- Train with Mall Dataset
- Develop GUI to facilitate testing and also estimate crowd count
- Predict set of images with or without ground truth
- Display a graph between ground truth and prediction
- Save a data file if need be
- Predict live video feed

### 1.3 Proposed Approach

The proposed approach of this project is to first create the neural network and train it. The development will be done using the PyTorch framework and also Python 3. Also, the practice of object-oriented programming will also be used throughout this project.



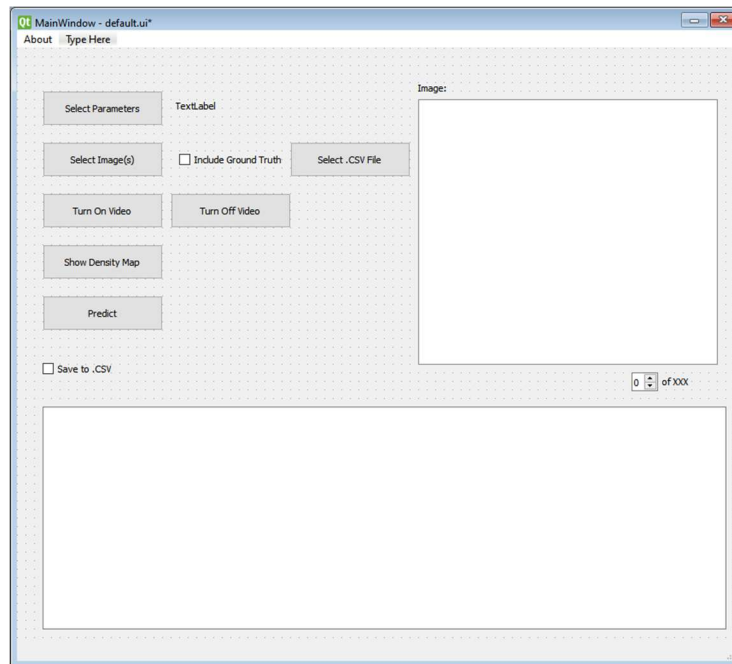
*Figure 1-3-1 Gaussian Density Map Formation*



*Figure 1-3-2 How train.py utilizes dataloader.py and model2.py together*

The neural network structure will be stored in model2.py, the code that will feed the image data will be stored in dataloader.py, the code to create the ground truth density map will be stored in gaussian.py and the file to train the neural network will be called as train.py.

Once the training of the neural network is complete and tested successfully, the development of the front-end can be initiated.



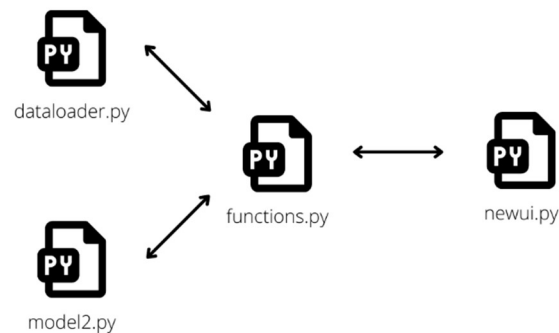
*Figure 1-3-3 Sketch of the GUI*



## Chapter 1: Introduction

The overall design of the GUI will be done using QtDesigner to help expedite the development time. After that, the GUI has to be connected to methods that will communicate with the neural network to form a working system that could estimate the crowd count and also, test the neural network for its accuracy.

To display the graphs and also the density map, the matplotlib Python library is used to visualize the tensor data.



*Figure 1-3-4 How the system files work with each other*

The user interface design will be stored in `newui.py` which will contain the GUI and also methods related to it. The methods that will help communicate with the GUI and the neural network structure however will be stored in `functions.py` for ease of development.

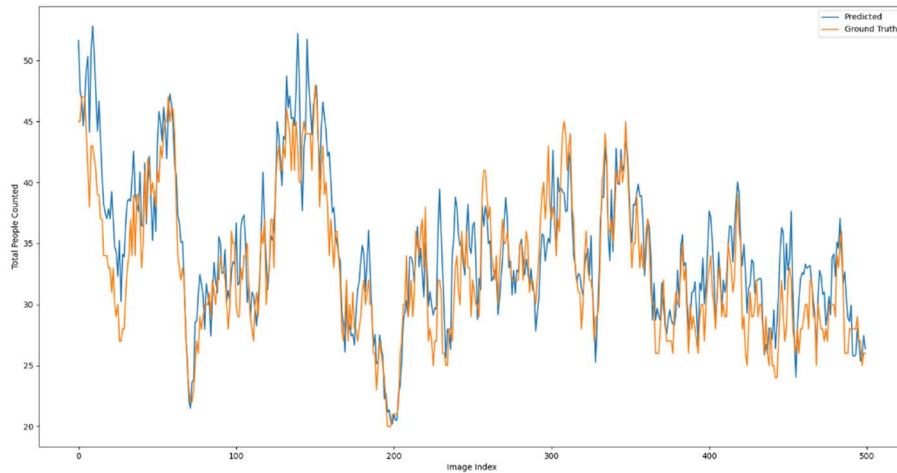
The overall code structure of this project will be split into several Python scripts for simplicity's sake. Moreover, the Pytorch structure is object-oriented based which makes developing the model much easier.

- *gaussian.py* – responsible for creating the ground truth density map
- *dataloader.py* – contains the Dataset class needed.
- *model.py* – contains the Network model class needed.
- *train.py* – responsible for implementing the Dataset and Network class together and train said model
- *functions.py* – contains methods that uses the neural network.

## Chapter 1: Introduction

- *newui.py* – main file to contains the UI and methods that calls methods from *functions.py*.

### **1.4 Highlight of What Have Been Achieved**



*Figure 1-4-1 Graph between Prediction and Ground Truth*

Mean Absolute Error: 2.45  
Mean Squared Error: 9.72  
Mean Inference Time: 12.97ms  
Total Duration: 6484.33ms

*Figure 1-4-2 MAE and MSE of the trained neural network*

The neural network that was trained achieved an accuracy of 2.45 in MAE and 9.72 in MSE when tested using the Test portion of the Mall dataset as show on the figure above.

## Chapter 1: Introduction

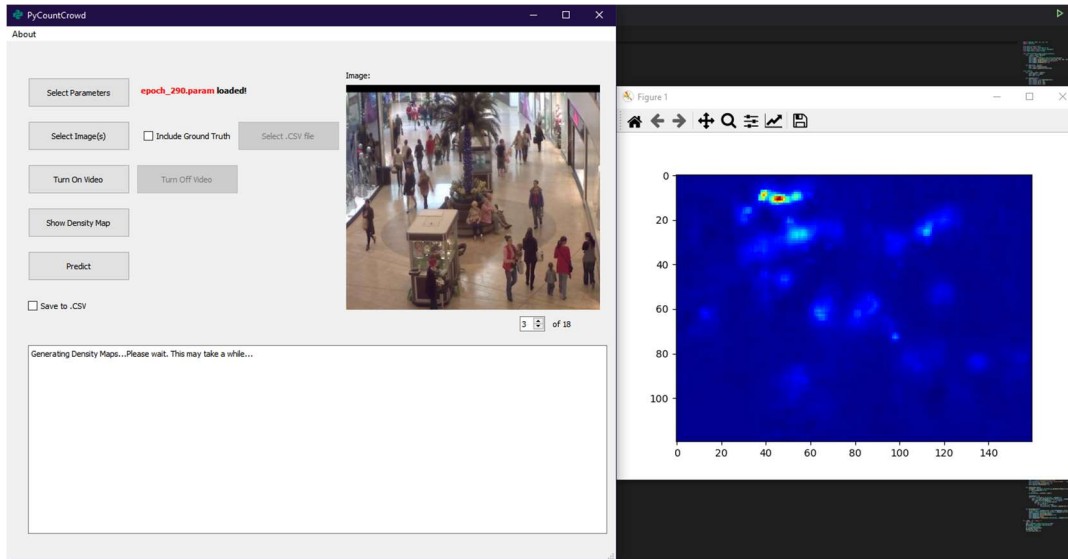


Figure 1-4-3 System displaying estimated density map

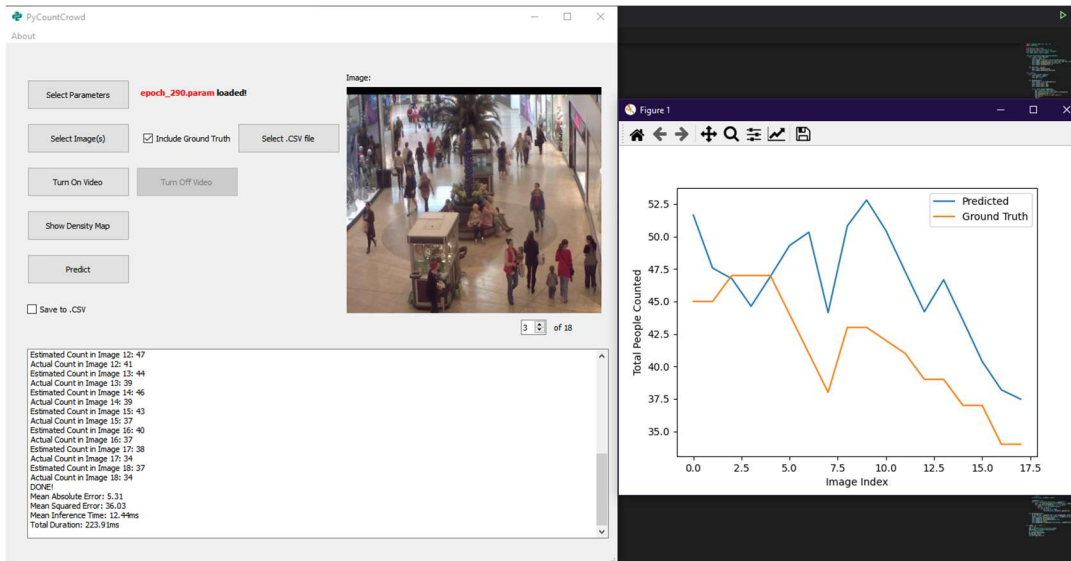


Figure 1-4-4 System displaying estimated crowd count and graph analysis

The front-end on the other hand, is also able to select the parameter file for the network and also, select multiple images to be displayed and viewed individually. Then, it could provide the estimated density maps of the image and also calculate the prediction of the crowd count by displaying the data in the log below. If needed, the predict button can also save those data into a csv file if the checkbox is ticked.

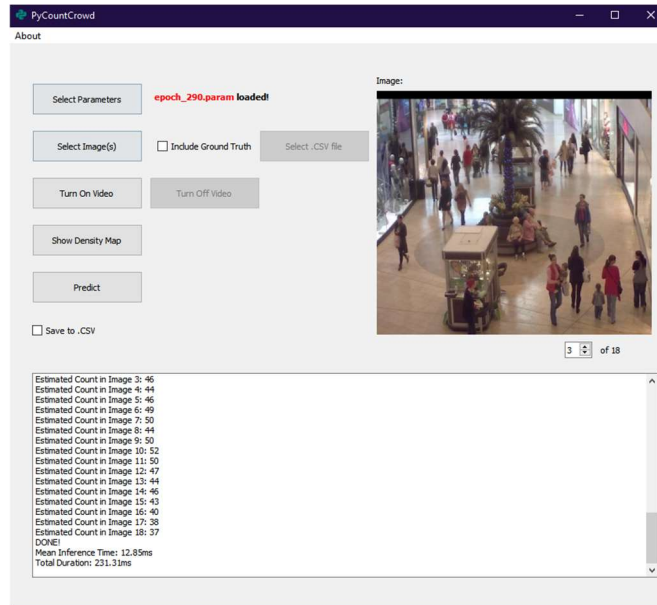


Figure 1-4-5 System displaying estimated crowd count without ground truth

The system can also predict without the ground truth file should it be unavailable for the images. Not only that it can also collect a live surveillance feed from a webcam to estimate the crowd as show in the figure below.

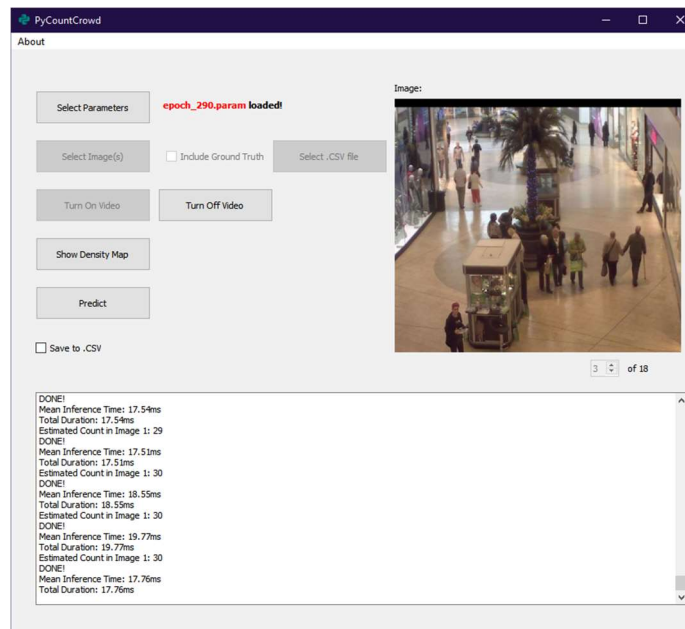


Figure 1-4-6 System displaying estimated crowd count of video feed

### **1.5 Report Organization**

This report is organized into 6 chapters, which are as follow:

- Chapter 1: Introduction
- Chapter 2: Literature Review
- Chapter 3: System Design
- Chapter 4: System Implementation
- Chapter 5: System Evaluation and Discussion
- Chapter 6: Conclusion

The first chapter will be the introduction of this project which includes problem statement and motivation, objectives, proposed approach, highlight of what have been achieved, and report organization. The second chapter is the literature review of the project, mainly about the neural network developed in the past and its strengths and weaknesses between them. The third chapter is reviewing the design of the system and how both front and back end of the system communicates with each other. The fourth chapter is discussing about how the whole project went out and what were the outcome and the project timeline. The fifth chapter is discussing about the testing of the system and ensure that it is in working order and also discusses about its disadvantages. The final chapter will be about the conclusion of this project, problems encountered, some personal insights as well as what future work could be done to this project.

## **CHAPTER 2: LITERATURE REVIEW**

Over the past years, numerous works have been done to solve the problem of crowd-counting as well as estimating crowd density using deep learning methods. In fact, there are numerous approaches that are proposed to achieve a successful crowd count. To make things simple, the methods are divided into three: Detection-based, Regression-based and Machine Learning based.

### *Detection-based Methods*

Detection methods are one of the earliest ways to count crowds. It is usually based on detecting an object in a monolithic form or based on a part of the human bodies.

In one study (Kong, et al., 2006), they developed a method to count the amount of people in a crowd from a single camera. Their method also considers feature normalization to deal with perspective projection and camera orientation. Not only that, instead of using simple features such as blob pixels or a summation of edge, feature histograms are adopted. With those two aspects, their system is trained to be viewpoint invariant and can be deployed with minimal setup. (Kong, et al., 2006) However, there is a lack of overcrowding in the training data which might cause some difficulties in counting in people in a dense crowd.

On the other hand, Dalal & Triggs (2005) developed a method that is based on assessing well-normalized local histograms of image gradient orientations in a dense grid whereby the local object appearance and shape can often be characterized quite properly by the distribution of local intensity gradients or edge directions even without precise knowledge of the corresponding gradient or edge positions. To implement this, they segment the images into several parts in which they called it as “cells” where each of them accrues a local 1-D histogram of gradient directions over the pixels of the cell. In that study, their method relies on a monolithic detection where a human body is detected by encircling them with a rectangular outline as shown in Figure 2-1.



*Figure 2-1: Sample images from their human detection database. (Dalal & Triggs, 2005)*

Monolithic detection works great on a sparse crowd as it can easily detect out each individual person out but unfortunately, it does not produce great results in a dense crowd as most of the people are severely hindered by occlusions. A remedy to this would be a part-based detection where a certain body part is detected instead of the whole.

Like the method stated previously, Li, et al. (2008) method also utilizes Histograms of Oriented Gradients based detection but instead of the whole body, it only detects from the head to the shoulder as shown in Figure 2-2 and not only that, it also combines a Mosaic Image Difference based foreground segmentation algorithm to provide an accurate estimation of crowd counting in a given area.



*Figure 2-2: Omega salient feature (left), sample images used in training set (right) (Li, et al., 2008)*

However, even with part-based detection method, it can still fall prey to generating inaccurate counts when the crowd gets denser but at least not as easily as compared to monolithic detection.

### Regression-based Methods

Regression-based methods are another form of detecting object. Where detection-based methods falters, regression-based methods shine. One of the major performance issues in detection-based crowd counting is when it tries to crowd count in images captured with low resolution as well as dense crowds and this is where this method performs better.

With that said, since this method focuses on the global characteristics of the crowd, it usually fares more well than traditional detection-based methods. This method usually learns a mapping between global and local features of an image that were extracted by operators, such as Scale-invariant feature transform (SIFT) and Local Binary Pattern (LBP). (Miao, et al., 2019)

For instance, Chan, et al. (2008) developed a system that counts the number of people walking in each direction while preserving their privacy with a Gaussian process regression. First, they segment the video into crowd regions that are moving in different directions by adopting the mixture of dynamic textures sequentially. Then, before the mixture models undergo feature extraction, Chan, et al. (2008) considered the effects of perspective as objects that are closer to the camera will appear larger which will cause any feature extraction from a foreground object to account for a smaller portion of the object when compared to the one extracted from an object that is further away. So, the video segments have to be normalized in a perspective manner before extracting features. Finally, a Gaussian process (GP) regression is used to regress feature vectors based on the number of people in each segment to count the total people in the crowd. A summary of this process is shown in Figure 2-3.

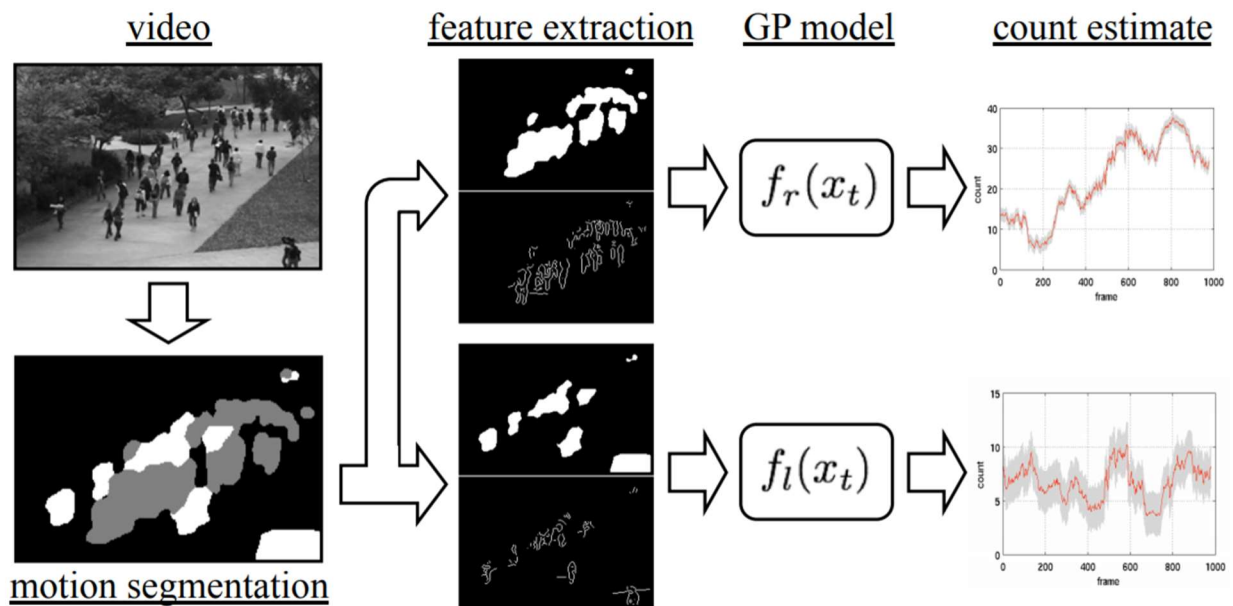


Figure 2-3 Outline of Process (Chan, et al., 2008)

Unfortunately, this method does come with a drawback. The spatial information of the people in the crowd is omitted so locating a specific person would be impossible which ironically is one of the objectives of the study as it, as the title of the study suggests, preserves their privacy.

On another note, there have been studies that incorporate both regression-based methods as well as detection based methods. Namely, DecideNet by Liu, et al. (2018) where it can adaptively decide which method to use to count in different locations based on its real density conditions with guidance of an attention mechanism. The results are profound as it is able to



provide accurate counts in three different datasets that have different crowd densities. (Liu, et al., 2018)

Kumagai, et al. (2018) also proposed a crowd counting method that uses multiple predictors that are dedicated to each particular appearance in which they are adaptively selected based on the appearances of the target. Unfortunately, this proposed method relies on vague training data and because of this, density map should be used to improve accuracy hence, more future work is needed.

However, by integrating both methods at the same time, these method would be computationally complex and also time-consuming to do so.

### Machine Learning Methods

With the growing popularity of machine learning in the past years, it is a no-brainer to make use of this innovation in crowd counting. In fact, it is one of the more popular methods than the hand-crafted methods so to speak.

On another hand, another approach was done by (Kanwal, et al., 2018) which involves an input coloured image that is converted to an enhanced image format that permits an improved feature extraction (the heads as foreground and the rest as background) done with Gaussian Mixture Model (GMM), an unsupervised machine learning technique which provides an estimate of the crowd density by analyzing blobs through connected components. The framework that they designed is divided into two modules where the first one is the foundation of the designed infrastructure dubbed as “Training” and the second module (Counting) which is responsible for counting by evaluating the output of the first module to estimate the key features in analyzed data. The experimental result was profound, it had a success rate of ninety percent, and it only took 19ms to detect a human in a frame proving its efficiency. (Kanwal, et al., 2018) However, the designed system is trained to detect specific features of interest such as the head in a photograph with a top-down view and heavily relies on the image resolution at hand. If it had a low resolution then, the accuracy drops significantly due to occlusion and clutter scene. Moreover, since the system is designed only to detect a part of the body specifically the head, it is very likely that it will provide unreliable results due to its shape.

## Chapter 2: Literature Review

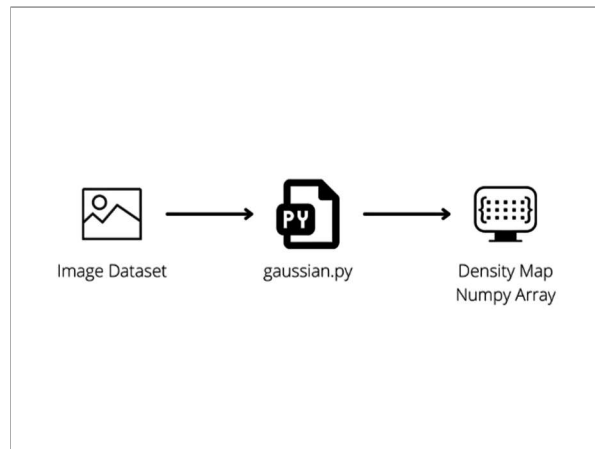
### Summary

With that said, it is without a doubt that proposing a crowd counting method with deep learning approach is the wisest choice as it is less time-consuming when compare to other approaches as well as not that complex to do so while producing considerable results.

**CHAPTER 3: SYSTEM DESIGN**

Before the system can be developed, a neural network needs to be trained. For that, a dataset has to be prepared. To prepare the dataset for the neural network, images from the dataset must be fed into a gaussian filter to create a density map in the form of NumPy array.

For this, gaussian.py is used where the annotated coordinates of the people's head in the image are applied a gaussian filter and the resulting map is saved as an .npy file.



*Figure 3-1 Gaussian Density Map Formation*

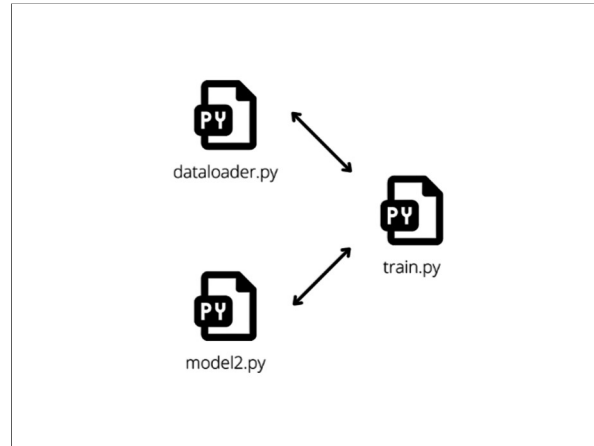
Once the ground truth density map is created, the training of the neural network can begin. First off, the dataset class from Pytorch is overridden in dataloader.py as the Mall Dataset is not native to PyTorch. This is where the images including the density maps are downsampled due to the use of MaxPool2d within the neural network so the getitem method needs to be overloaded. Dataloader.py is responsible for retrieving the images from the Dataset and feeding it into the neural network to be used for training.

Then, the neural network is located in the model2.py, the neural network structure is similar to the neural network structured mentioned in the paper by (Zhang, et al., 2016) except an additional layer of batch normalization (BatchNorm2d) is added after the first convolution 2d (Conv2d) layer for each of the neural network column.

```

self.pipeline1=nn.Sequential(
    nn.Conv2d(in_channels=3, out_channels=16, kernel_size=9, padding=4),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(2),
    nn.BatchNorm2d(16),
  
```

*Figure 3-2 Code snippet of the neural network model*



*Figure 3-3 How train.py utilizes dataloader.py and model2.py together*

Finally, the neural network can be trained using `train.py`. This file will call and initialize both the classes within the `dataloader.py` and `model2.py`. The neural network is then made to be processed by the GPU instead of the CPU. The loss function used here is the `MSELoss` from the PyTorch library and the optimizer that is used is the Adam optimizer with a learning rate of 0.00146 and `AMSGrad` set as enabled. Then, the dataloader for testing and validation with a batch size of 16 is initiated. If the GPU has Tensor cores present, the training time can be expedited using PyTorch's native automatic mixed precision function which in this project's case, is used. The neural network then trains with the training dataset and then validated using validation dataset to evaluate the loss and accuracy. This iteration repeats until either a sufficient accuracy is reached, or the set amount of epoch has reached. The network parameters are saved every time the neural network achieves a lower loss. Finally, the neural network is then given an input within the dataset that it has never seen before during training to calculate its true accuracy.

Once the neural network training and testing has been completed, the development of the GUI can be completed. By using QtDesigner, the construction of the GUI can be expedited by simply dragging and dropping the needed widgets into the window. The rough sketch of the system's GUI can be seen in the figure below.

newui.py

The required libraries and imports are shown in the figure below.

```
import logging, math, os, csv, cv2
import functions

from pathlib import Path
from datetime import datetime as dt
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import QTimer
```

Figure 3-4 Code snippet of newui.py

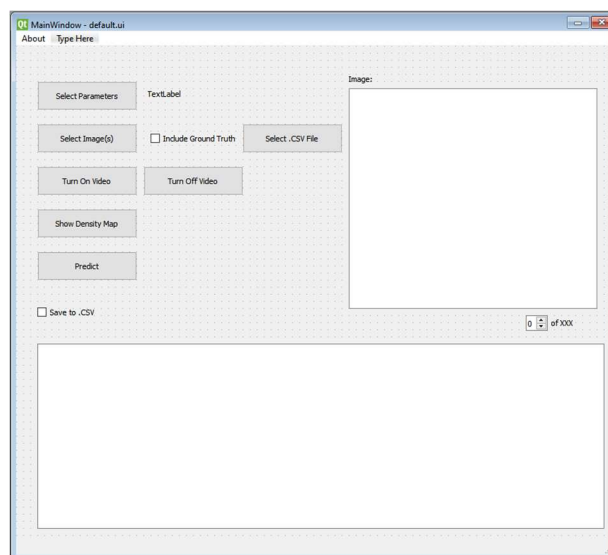


Figure 3-5 Sketch of the GUI

Of course, the sketch only has the needed widgets within the GUI and the connections still has be done manually. To start adding the connections, the UI file must first be converted into a PY file by using the Windows command terminal and entering the following command, with default.ui as the UI file name and output.py as the desired output file name as example:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.
C:\Users\mwafi>pyuic5 default.ui -o output.py
```

Figure 3-6 Command line to convert .ui to .py

### Select Parameters Button

The button once clicked, will open a file dialog for a .param file to be selected. Once selected, the .param file directory is saved into the \_\_paramFile variable and a textLabel is modified to show which .param file is loaded as shown in the figure below.

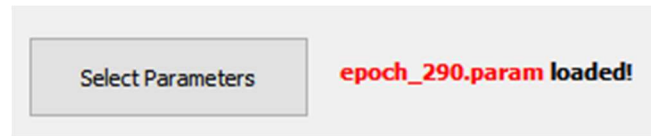


Figure 3-7 Image snippet of the system

```
self.selectNN = QtWidgets.QPushButton(self.centralwidget)
self.selectNN.setGeometry(QtCore.QRect(30, 50, 141, 41))
self.selectNN.setObjectName("selectNN")
self.selectNN.clicked.connect(self.paramFile)
```

Figure 3-8 Code snippet of newui.py regarding its connection to method

```
def paramFile(self):
    paramFile = QtWidgets.QFileDialog.getOpenFileName(filter="Parameter (*.param)")
    if len(paramFile[0]) == 0:
        return
    Ui_MainWindow.__paramFile = paramFile[0]
    text = os.path.basename(paramFile[0])
    self.paramTitle.setText("<b><FONT COLOR=#ff0000>" + text + "</FONT> loaded!" + "</b>")
    self.selectImage.setEnabled(True)
    self.openCam.setEnabled(True)
```

Figure 3-9 Code snippet of newui.py regarding the aforementioned method

After that, the buttons for selecting image and turning on video camera is enabled.

### Select Images Button

This button is responsible for selecting the images that needs to be processed by the neural network. It will also open a file dialog where the files can be selected. After that, the directory of the files is saved into an array called \_\_ImageArr.

```
self.selectImage = QtWidgets.QPushButton(self.centralwidget)
self.selectImage.setGeometry(QtCore.QRect(30, 110, 141, 41))
self.selectImage.setObjectName("selectImage")
self.selectImage.clicked.connect(self.openFile)
self.selectImage.setEnabled(False)
```

Figure 3-10 Code snippet of the select image button

```
def openFile(self):
    fileName = QtWidgets.QFileDialog.getOpenFileNames(filter="Images (*.png *.jpg *.bmp)")
    if len(fileName[0]) == 0:
        return
    self.showHeat.setEnabled(True)
    self.Predict.setEnabled(True)
    Ui_MainWindow.__ImageArr.clear()
    for x in fileName[0]:
        Ui_MainWindow.__ImageArr.append(x)
    if self.inclGT.isChecked():
        self.openFolder()
    Ui_MainWindow.__ImageCurrent = 0
    newpix = QtGui.QPixmap(Ui_MainWindow.__ImageArr[Ui_MainWindow.__ImageCurrent])
    self.ImageView.setPixmap(newpix)
    self.ImageView.setScaledContents(True)
    self.ImageView.show()
    self.imagecountLabel.setText("of " + str(len(Ui_MainWindow.__ImageArr)))

    self.ImageNumber.setValue(Ui_MainWindow.__ImageCurrent + 1)
    self.ImageNumber.setEnabled(True)
    self.ImageNumber.setRange(1, len(Ui_MainWindow.__ImageArr))
```

*Figure 3-11 Code snippet of the method connected to select image button*

Once the button is selected, it will enable the Show Density Map and Predict button. It will also clear any original file directories within the array for the new one to take its place. If the ground truth checkbox is ticked, it will also prompt another file dialog for the new ground truth file to be selected. After that, it sets the first image to be displayed onto the system. The images can be traversed using the spinbox below the display.

### Include Ground Truth Checkbox

If this checkbox is ticked, it will let the system know that a ground truth file is needed and enables the ground truth button for it to open a file dialog to select the ground truth file.

```
self.inclGT = QtWidgets.QCheckBox(self.centralwidget)
self.inclGT.setGeometry(QtCore.QRect(190, 120, 131, 21))
self.inclGT.setObjectName("inclGT")
self.inclGT.stateChanged.connect(lambda: self.selectGT.setEnabled(not (self.selectGT.isEnabled())))
```

*Figure 3-12 Code snippet of the include ground truth checkbox*

The checkbox does not connect to any method but instead uses a lambda function instead where it inverts the current settings for the Select Ground Truth (Select .CSV File) button.

### Select .CSV File

This is the ground truth button that was mentioned earlier. Only .CSV files are allowed for the system with the first column as the image file name and second column as the ground truth count.

```

self.selectGT = QtWidgets.QPushButton(self.centralwidget)
self.selectGT.setGeometry(QtCore.QRect(320, 110, 141, 41))
self.selectGT.setObjectName("selectGT")
self.selectGT.clicked.connect(self.openFolder)
self.selectGT.setEnabled(False)

```

Figure 3-13 Code snippet of the select .csv file

```

def openFolder(self):
    fileName = QtWidgets.QFileDialog.getOpenFileName(filter=".CSV (*.csv)")
    if len(fileName[0]) == 0:
        return
    Ui_MainWindow.__GrndTArr.clear()

    imageNames = []
    for x in range(len(Ui_MainWindow.__ImageArr)):
        name = os.path.basename(Path(Ui_MainWindow.__ImageArr[x]))
        with open(Path(fileName[0]), 'r') as gtcsv:
            gtcsv_r = csv.DictReader(gtcsv)
            for row in gtcsv_r:
                if row['name'] == name:
                    Ui_MainWindow.__GrndTArr.append(row['count'])

```

Figure 3-14 Code snippet of the method connected to the select .csv file button

If there are no files selected, it will return without doing anything. If there is a file selected, the ground truth array, \_\_GrndTArr is cleared for the new ground truth to take place.

It will first traverse through the \_\_ImageArr to get the filename and then, it will compare that said filename in the .CSV file to retrieve its ground truth and append it to the \_\_GrndTarr array.

### Turn On Video / Turn Off Video Button

Clicking the Turn On Video button, displays the webcam live feed and captures the video image and processed by the neural network for it to get predicted. Turn Off Video button will disable the webcam and go back to image processing mode.

```

self.openCam = QtWidgets.QPushButton(self.centralwidget)
self.openCam.setGeometry(QtCore.QRect(30, 170, 141, 41))
self.openCam.setObjectName("openCam")
self.openCam.clicked.connect(self.countCam)
self.openCam.setEnabled(False)

self.closCam = QtWidgets.QPushButton(self.centralwidget)
self.closCam.setGeometry(QtCore.QRect(180, 170, 141, 41))
self.closCam.setObjectName("openCam")
self.closCam.clicked.connect(self.stopCam)
self.closCam.setEnabled(False)

```

Figure 3-15 Code snippet of the method connected to the turn on video / turn off video button



```
self.camera = Camera(1)

self.timer = QTimer()
self.timer.timeout.connect(self.updateCam)
```

Figure 3-16 Camera and QTimer instantiation

```
class Camera:
    def __init__(self, camera):
        self.camera = camera
        self.cap = None

    def openCam(self):
        self.vidcam = cv2.VideoCapture(1)
        self.vidcam.set(5, 30)
        self.vidcam.set(3, 640)
        self.vidcam.set(4, 480)

        if not self.vidcam.isOpened():
            msg = QtWidgets.QMessageBox()
            msg.setIcon(QtWidgets.QMessageBox.Information)
            msg.setWindowTitle("Error")
            msg.setText("Failed to Open Camera!")
            msg.exec_()
            return

    def stopCam(self):
        self.vidcam.release()
        cv2.destroyAllWindows()

    def initialize(self):
        self.cap = cv2.VideoCapture(self.camera)
```

Figure 3-17 Camera class

```
def countCam(self):
    self.camera.openCam()
    _, frame = self.camera.vidcam.read()
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    image = QtGui.QImage(frame, frame.shape[1], frame.shape[0], QtGui.QImage.Format_RGB888)
    pixmap = QtGui.QPixmap.fromImage(image)
    self.ImageView.setPixmap(pixmap)
    self.timer.start(int(1000 / 20))
    logging.info("Camera opened!")
    self.closCam.setEnabled(True)
    self.openCam.setEnabled(False)
    self.selectImage.setEnabled(False)
    self.inclGT.setEnabled(False)
    self.ImageNumber.setEnabled(False)
    return

def updateCam(self):
    _, frame = self.camera.vidcam.read()
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    image = QtGui.QImage(frame, frame.shape[1], frame.shape[0], QtGui.QImage.Format_RGB888)
    pixmap = QtGui.QPixmap.fromImage(image)
    param = Path(Ui_MainWindow.__paramFile)
    ne_count, curr_time = functions.predictV(frame, param)
    for i in range(len(ne_count)):
        logging.info("Estimated Count in Image " + str(i + 1) + ": " + str(math.trunc(ne_count[i])))
    logging.info("DONE!")
    logging.info("Mean Inference Time: " + str("{:.2f}".format(curr_time)) + "ms")
    logging.info("Total Duration: " + str("{:.2f}".format(curr_time * len(ne_count))) + "ms")
    self.ImageView.setScaledContents(True)
    self.ImageView.setPixmap(pixmap)
    return

def stopCam(self):
    self.camera.stopCam()
    self.timer.stop()
    self.closCam.setEnabled(False)
    self.openCam.setEnabled(True)
    self.selectImage.setEnabled(True)
    self.inclGT.setEnabled(True)
    self.ImageNumber.setEnabled(True)
```

Figure 3-18 Methods responsible for the live video feed functionality

The video feed is captured as image every 50ms and fed into the neural network for it to get the predicted crowd count.

By clicking Turn On Video, the countCam method is initiated. It will first read the frame from the video feed and then converted to RGB format using CV2 and displayed onto the system.

A timer of 50ms using QTimer is made and whenever this time reaches, updateCam method is called and frame is again read from the video feed and displayed onto the system using the same sequence as before but this time, the frame is sent to neural network for it get the predicted crowd count.

By clicking the Turn Off Button, it disables all the camera and timer and reverts back to image processing mode.

### Show Density Map Button

Clicking this button will display the current image's estimated density map in another window by calling another method in functions.py.

```
self.showHeat = QtWidgets.QPushButton(self.centralwidget)
self.showHeat.setGeometry(QtCore.QRect(30, 230, 141, 41))
self.showHeat.setObjectName("showHeat")
self.showHeat.clicked.connect(self.estimateDmap)
self.showHeat.setEnabled(False)
```

*Figure 3-19 Code snippet of the show density map button*

```
def estimateDmap(self):
    if (len(Ui_MainWindow.__ImageArr) == 0):
        return
    logging.info("Generating Density Maps...Please wait. This may take a while...")
    image = Ui_MainWindow.__ImageArr[Ui_MainWindow.__ImageCurrent]
    param = Path(Ui_MainWindow.__paramFile)
    functions.estimate_density_map(image, param)
```

*Figure 3-20 Code snippet of the method connected to the button*

Predict Button

Clicking this button will predict the selected images and if the ground truth checkbox is ticked, it will produce a graph analysis of the estimated count against the ground truth and also, the accuracy of the neural network in MAE and MSE.

```
self.Predict = QtWidgets.QPushButton(self.centralwidget)
self.Predict.setGeometry(QtCore.QRect(30, 290, 141, 41))
self.Predict.setObjectName("Predict")
self.Predict.clicked.connect(self.runScript)
self.Predict.setEnabled(False)
```

Figure 3-21 Code snippet of the predict button

```
def runScript(self):
    logging.info("Please wait. This may take a while...")
    if self.selectGT.isEnabled():
        images = Ui_MainWindow.__ImageArr
        gtfile = Ui_MainWindow.__GrndTArr
        param = Path(Ui_MainWindow.__paramFile)
        mae, mse, ne_count, curr_time = functions.graph(images, gtfile, param)
        for i in range(len(ne_count)):
            logging.info("Estimated Count in Image " + str(i + 1) + ": " + str(math.trunc(ne_count[i])))
            logging.info("Actual Count in Image " + str(i + 1) + ": " + str(gtfile[i]))
        logging.info("DONE!")
        logging.info("Mean Absolute Error: " + str("{:.2f}".format(mae)))
        logging.info("Mean Squared Error: " + str("{:.2f}".format(mse)))
        logging.info("Mean Inference Time: " + str("{:.2f}".format(curr_time)) + "ms")
        logging.info("Total Duration: " + str("{:.2f}".format(curr_time*len(images))) + "ms")
    else:
        images = Ui_MainWindow.__ImageArr
        param = Path(Ui_MainWindow.__paramFile)
        ne_count, curr_time = functions.predict(images, param)
        for i in range(len(ne_count)):
            logging.info("Estimated Count in Image " + str(i + 1) + ": " + str(math.trunc(ne_count[i])))
            logging.info("DONE!")
        logging.info("Mean Inference Time: " + str("{:.2f}".format(curr_time)) + "ms")
        logging.info("Total Duration: " + str("{:.2f}".format(curr_time*len(images))) + "ms")
    if (self.checkCSV.isChecked()):
        folder = dt.now()
        folder = folder.strftime("%d-%m-%Y %H-%M-%S")
        os.mkdir(str(folder))
        file = os.path.join(folder, "data.csv")
        with open(file, mode="w") as dfile:
            keynames = ["name", "estimated"]
            dfilew = csv.DictWriter(dfile, fieldnames=keynames)
            dfilew.writeheader()
            for x in range(len(images)):
                dfilew.writerow({"name" : str(os.path.basename(images[x])), "estimated" : str(math.trunc(ne_count[x]))})
        file = os.path.join(folder, "data.txt")
```

Figure 3-22 Code snippet of the method connected to the predict button

The button will first see if the ground truth checkbox is ticked and if it does, it will run a different method in functions.py and if it does not, it will run another method with only the estimation output in functions.py.

If the Save to .CSV is ticked, it will also save the data into a .csv file within a created folder with the current date and time.

Functions.py

The required libraries and imports are shown in the figure below.

```
import torch
import matplotlib.pyplot as plt
import numpy as np
import cv2

from numpy import asarray, savetxt
from model2 import Network
from dataloader import MallDataset
from matplotlib import cm as CM
from pathlib import Path
```

Figure 3-23 Code snippet of the required libraries and imports

def predict(images, param)

```
def predict(images, param):
    device = torch.device("cuda")
    net = Network().to(device)
    net.load_state_dict(torch.load(Path(param)))
    img_root='..\mall_dataset\\test_dataset\\1'
    gt_dmap_root='..\mall_dataset\\ground_truth'
    dataset = MallDataset(img_root, gt_dmap_root, 4)
    dataloader = torch.utils.data.DataLoader(dataset, batch_size=1, shuffle=False)
    net.eval()
    starter, ender = torch.cuda.Event(enable_timing=True), torch.cuda.Event(enable_timing=True)
    timings = np.zeros((len(images), 1))
    e_count = np.array([])

    with torch.no_grad():
        for x, (tens, gt_dmap) in enumerate(dataloader):
            if (x == (len(images))):
                break
            img = plt.imread(Path(images[x]))
            row = int(img.shape[0] // 4)
            col = int(img.shape[1] // 4)
            img = cv2.resize(img, (col * 4, row * 4))
            img = img.transpose((2, 0, 1))
            img_tensor = torch.tensor(img, dtype = torch.float)
            img_tensor = img_tensor.unsqueeze(0)
            img_tensor = img_tensor.to(device)
            gt_dmap = gt_dmap.to(device)
            starter.record()
            et_dmap = net(img_tensor).detach()
            ender.record()
            torch.cuda.synchronize()
            curr_time = starter.elapsed_time(ender)
            timings[x] = curr_time
            e_count = np.append(e_count, et_dmap.cpu().data.sum().item())
    return asarray(e_count), (np.sum(timings) / len(images))
```

Figure 3-24 predict method that instantiates the neural network to predict count of crowd images

When this method is called along with the parameters, it will initiate the neural network model with the given images array and the parameter file. A Torch Event timer is also used to calculate



the inference time. Note that the network is set into evaluation mode and with that, the images are fed into the neural network and the estimated count array is output into `e_count`.

The `e_count` numpy array is then returned as a normal array and the average inference time is also returned.

*`def predictV(images, param)`*

```
def predictV(images, param):
    device = torch.device("cuda")
    net = Network().to(device)
    net.load_state_dict(torch.load(Path(param)))
    img_root='.\\mall_dataset\\test_dataset\\1'
    gt_dmap_root='.\\mall_dataset\\ground_truth'
    dataset = MallDataset(img_root, gt_dmap_root, 4)
    dataloader = torch.utils.data.DataLoader(dataset, batch_size=1, shuffle=False)
    net.eval()
    starter, ender = torch.cuda.Event(enable_timing=True), torch.cuda.Event(enable_timing=True)
    timings = np.zeros((1, 1))
    e_count = np.array([])

    with torch.no_grad():
        for x, (tens, gt_dmap) in enumerate(dataloader):
            img = images
            row = int(img.shape[0] // 4)
            col = int(img.shape[1] // 4)
            img = cv2.resize(img, (col * 4, row * 4))
            img = img.transpose((2, 0, 1))
            img_tensor = torch.tensor(img, dtype = torch.float)
            img_tensor = img_tensor.unsqueeze(0)
            img_tensor = img_tensor.to(device)
            gt_dmap = gt_dmap.to(device)
            starter.record()
            et_dmap = net(img_tensor).detach()
            ender.record()
            torch.cuda.synchronize()
            curr_time = starter.elapsed_time(ender)
            timings[x] = curr_time
            e_count = np.append(e_count, et_dmap.cpu().data.sum().item())
            break
    return asarray(e_count), (np.sum(timings) / 1)
```

*Figure 3-25 method of `predictV` that predicts the live video feed count*

When this method is called along with the parameters, it will initiate the neural network model with the given input image and parameter file. A Torch Event timer is also used to accurately calculate the inference time. Since the image is already in NumPy array form, no conversion is needed, and it is directly fed into the neural network. Even though only one image was input, the return is still in array form so that it will remain working with the code that pairs with the previous method. The inference time is also returned.

*def estimate\_density\_map(image, param)*

```
def estimate_density_map(image, param):
    device = torch.device("cuda")
    net = Network().to(device)
    net.load_state_dict(torch.load(Path(param)))
    img_root='.\mall_dataset\\test_dataset\\1'
    gt_dmap_root='.\mall_dataset\\ground_truth'
    dataset = MallDataset(img_root, gt_dmap_root, 4)
    dataloader = torch.utils.data.DataLoader(dataset, batch_size=1, shuffle=False)
    net.eval()

    with torch.no_grad():
        for i, (_, _) in enumerate(dataloader):
            img = plt.imread(Path(image))
            row = int(img.shape[0] // 4)
            col = int(img.shape[1] // 4)
            img = cv2.resize(img, (col * 4, row * 4))
            img = img.transpose((2, 0, 1))
            img_tensor = torch.tensor(img, dtype = torch.float)
            img_tensor = img_tensor.unsqueeze(0)
            img_tensor = img_tensor.to(device)
            et_dmap = net(img_tensor).detach()
            et_dmap = et_dmap.squeeze(0).squeeze(0).cpu().numpy()
            plt.figure()
            plt.imshow(et_dmap, cmap=CM.jet)
            plt.show()
            break
    return
```

*Figure 3-26 estimate\_density\_map method code snippet*

When this method is called along with the parameters, the neural network model will be initiated with the given input image and parameter file. Since it just requires the estimated density map, none of the data from dataloader is needed hence, the underscores. When done, it displays the density map and returns to the caller.

*def graph(image, gtArr, param)*

```
def graph(image, gtArr, param):
    device = torch.device("cuda")
    net = Network().to(device)
    net.load_state_dict(torch.load(Path(param)))
    img_root = '\\mall_dataset\\test_dataset\\1'
    gt_dmap_root = '\\mall_dataset\\ground_truth'
    dataset = MallDataset(img_root, gt_dmap_root, 4)
    dataloader = torch.utils.data.DataLoader(dataset, batch_size=1, shuffle=False)
    net.eval()
    starter, ender = torch.cuda.Event(enable_timing=True), torch.cuda.Event(enable_timing=True)
    timings = np.zeros((len(image), 1))
    mae = 0
    mse = 0
    e_count = np.array([])
    ne_count = np.array([])
    count = np.array([])

    with torch.no_grad():
        img_len = np.arange(len(image))
        for i, (_, _) in enumerate(dataloader):
            if (i == (len(image))):
                break
            img = plt.imread(Path(image[i]))
            row = int(img.shape[0] // 4)
            col = int(img.shape[1] // 4)
            img = cv2.resize(img, (col * 4, row * 4))
            img = img.transpose((2, 0, 1))
            img_tensor = torch.tensor(img, dtype = torch.float)
            img_tensor = img_tensor.unsqueeze(0)
            img_tensor = img_tensor.to(device)

            starter.record()
            et_dmap = net(img_tensor).detach()
            ender.record()
            torch.cuda.synchronize()
            curr_time = starter.elapsed_time(ender)
            timings[i] = curr_time
            mae += abs(et_dmap.cpu().data.sum() - int(gtArr[i])).item()
            mse += pow(abs(et_dmap.cpu().data.sum() - int(gtArr[i])), 2)
            e_count = np.append(e_count, et_dmap.cpu().data.sum().item())
            count = np.append(count, int(gtArr[i]))

    plt.plot(img_len, e_count, label = "Predicted")
    plt.plot(img_len, count, label = "Ground Truth")
    plt.xlabel("Image Index")
    plt.ylabel("Total People Counted")
    plt.legend()
    plt.show()
    return (mae/len(image)), (mse/len(image)), asarray(e_count), (np.sum(timings) / len(image))
```

Figure 3-27 graph method that displays the graph figure, MAE and MSE

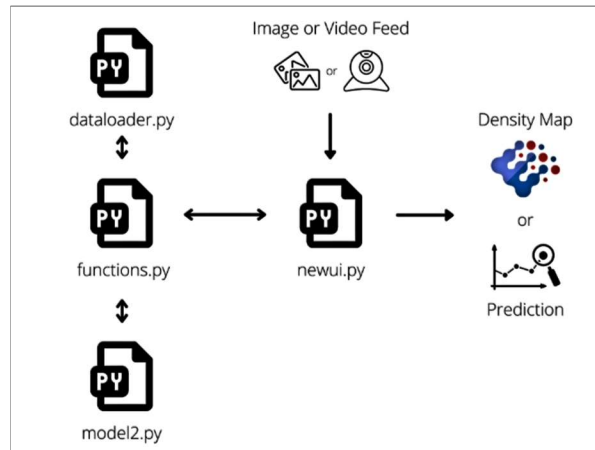
When this method is called along with the parameters, the neural network model will be initiated with the given input image, ground truth and parameter file. A Torch Event timer is also used to calculate the inference time. Note that the network is set into evaluation mode and with

## Chapter 3: System Design

that, the images are fed into the neural network and the estimated count array is output into `e_count`. The MAE and MSE is also calculated at every iteration of the images. The graph of estimated count against ground truth is also displayed in a window.

The `e_count` numpy array is then returned as a normal array and the average inference time, MAE and MSE is also returned.

### Overall Design



*Figure 3-28 Diagram of how the system works*

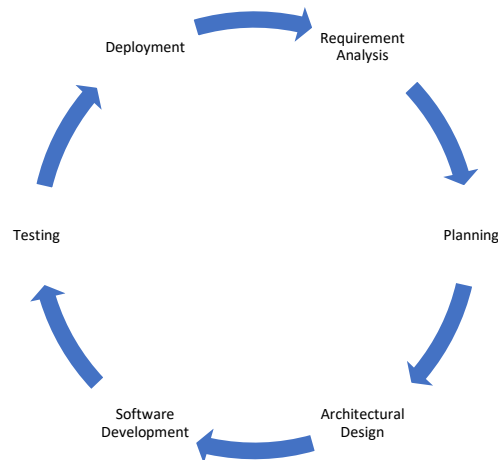
`newui.py` which is the main file gets input of image or video feed as well as the parameter file and sends those data into methods located in `functions.py` which utilizes the `dataloader.py` and `model2.py` to process using the neural network and outputs back the necessary data such as density map or crowd counting prediction into `newui.py`.



## **CHAPTER 4: SYSTEM IMPLEMENTATION**

### Methodology

In this project, the methodology used is called the Software Design Life Cycle (SDLC). The process is in a circular motion with 6 steps as shown in the figure below.



*Figure 4-1 SDLC Process Flowchart*

Since the problem has already been identified prior, the next stage would be the Planning stage where the necessary requirements need to be found and the expected timeline of the whole process is charted. After that, the next stage is the Design phase where a rough outline of the design is made and is reviewed thoroughly. When that is done, the development process starts which means the coding and research begins. This process would be the one that takes the longest, so careful planning is definitely needed beforehand. Next, when the model is all and done, it is then tested for any bugs or unintended outcomes. It's basically the debugging process. At the end of the first loop, the model is working but will still need some more fine tuning or updates in the process so, the model continues to get worked on at the forthcoming loop until there isn't much more to update.

### Tools

For this project, the tools that are needed are as follows:

- A working computer equipped with a CUDA-supported discrete graphics processing unit (GPU)
  - Preferably an NVIDIA card with RT cores
    - Zotac RTX 2060 Twin Fan is used for this project
  - At least 16 GB of RAM

## Chapter 4: System Implementation

- A CPU with at least 4 cores and 8 threads
- Pytorch machine learning library
- Python 3 and above
- Microsoft Visual Studio Code
  - For coding and debugging purposes
- Package installer for Python
  - Pip is used for this project
- Several external python libraries
  - scipy – Python library for scientific computing
  - numpy – to be used along with Pytorch
  - matplotlib – to read .mat files or visualization needs
  - pillow – Python imaging library
  - cv2 – OpenCV library for computer vision
  - PyQt5 – GUI toolkit
  - PyQt5Designer – GUI designer tool

### System Performance Definition

The improvements of the neural network model will be measured in mean absolute error as well as mean squared error.

Mean Absolute Error is an evaluation tool used for regression models where the mean absolute error of a model in terms of its test set is the mean of the absolute values of the individual prediction errors on every instance in the test set. So, the difference between the true value and the predicted value for the instance is the prediction error. (Sammut & Webb, 2011) The targeted MAE for this model is to be as low as possible.

$$mae = \frac{\sum_{i=1}^n abs(y_i - \lambda(x_i))}{n}$$

*Figure 4-2 Mean Absolute Error Equation (Sammut & Webb, 2011)*

Mean Squared Error on the other hand is also an evaluation tool used for regression models. The mean square error of the model in terms of its test set is the mean of the squared prediction error

## Chapter 4: System Implementation

on all the instances of the test set. The difference between the true value and the predicted value of the instance is the prediction error. (Sammut & Webb, 2011) The targeted MSE for this model is to be as low as possible.

$$mse = \frac{\sum_{i=1}^n (y_i - \lambda(x_i))^2}{n}$$

*Figure 4-3 Mean Squared Error Equation (Sammut & Webb, 2011)*

### Verification Plan

To verify that the model is working and accurate, a Python script will be used to measure the estimated crowd count by the neural network model against the ground truth which will be the actual number of crowds within the test set. From there, the MAE and MSE is calculated.

To verify the GUI, a table of test plan is made to verify that the intended actions are working.

### System Overview

The crowd counting algorithm will be built on a machine library framework called PyTorch. It is an open-source library developed by Facebook's AI Research Lab (FAIR) meant for applications such as computer vision and natural language processing.

The model will be based on a neural network published on a paper by Zhang et al. called Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. The neural network has three columns of a neural network that will be fused together at the end as shown in the figure below.

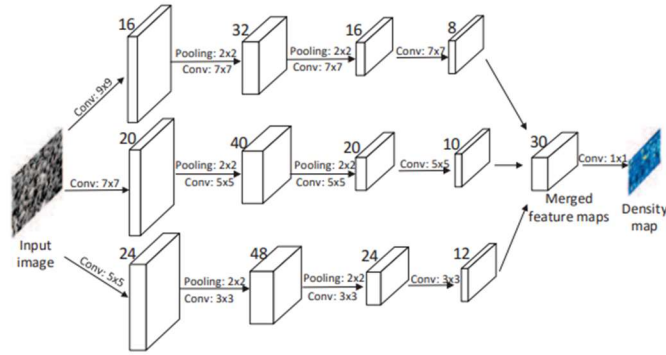


Figure 4-4 Structure of the proposed Multi-Column Convolutional Neural Network (Zhang, et al., 2016)

However, in comparison with the paper, the model in this project will be trained with the Mall Dataset instead of the Shanghai Tech Dataset and will be fine-tuned for a much sparser crowd as counting a dense crowd would require much more knowledge and research than a typical bachelor’s degree student would have.

Beforehand, the dataset that I will be using is annotated with coordinates of each of the person’s head in the instances of images within the dataset so that a ground truth of the actual number of a person is given. Then, these set of images are then applied with a gaussian kernel filter where the people’s head are located at to develop a density map ground truth for the neural network to work with.

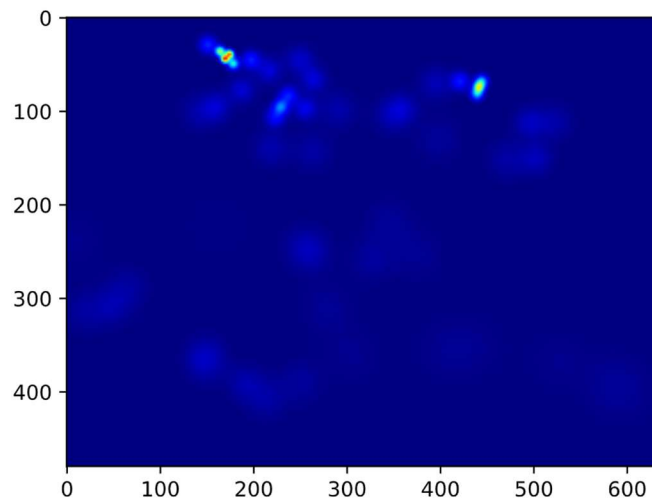
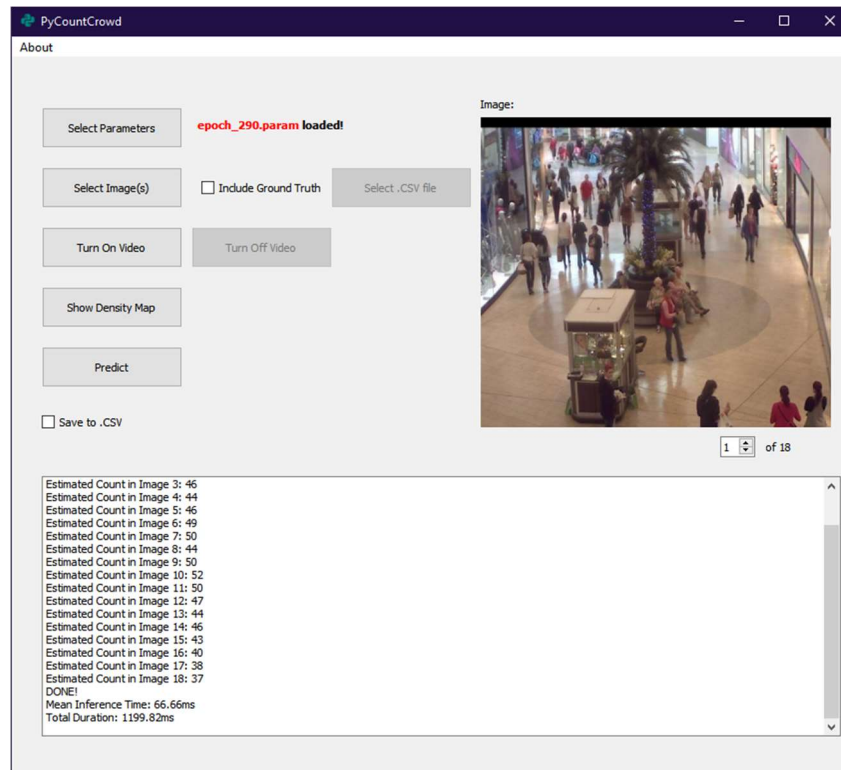


Figure 4-5 Example of the ground truth density map

Then, the dataset is then split into three groups, Training, Validation and Testing. The Training group are the ones that will be fed into the neural network for training and the Validation group is there so that the neural network model is not overfitted with the training dataset. To put in simpler terms, overfitting is when a neural network model is modeled with the training data excessively. The minor and incessant details of the training data are picked up and learned as concepts by the model where these concepts are not present or are irrelevant with new data and thus, hampering the model's ability to generalize. (Brownlee, 2019) Once the training is done or when there aren't many changes after each epoch, the model is then tested with the Testing data which it has never seen before to measure its accuracy. The image datasets will be fed as a single image instead of splitting the images as the crowd in the images are very sparse.

Back to the neural network model, according to the paper, it contains three parallel CNNs in which their filter contains the same network structure, but layers of batch normalization is added on each CNNs. (Zhang, et al., 2016) This CNN is using a 2 x 2 region max pooling to reduce the computations needed by the GPU and so the dataset would need to be down sampled beforehand. As for the activation function, which is what determines whether the neurons within the model are activated based on the calculated weighted sum and bias, Rectified Linear unit (ReLU) is used. This is because they are much more efficient, generalizes better and converge faster for deep learning networks such as this. (Zeiler, et al., 2013) Also, the optimizer used for this model is the Adam optimization algorithm which is already built into the Pytorch library. As mentioned before, to measure the accuracy of the trained model, MAE and MSE is used.



*Figure 4-6 Image of the system*

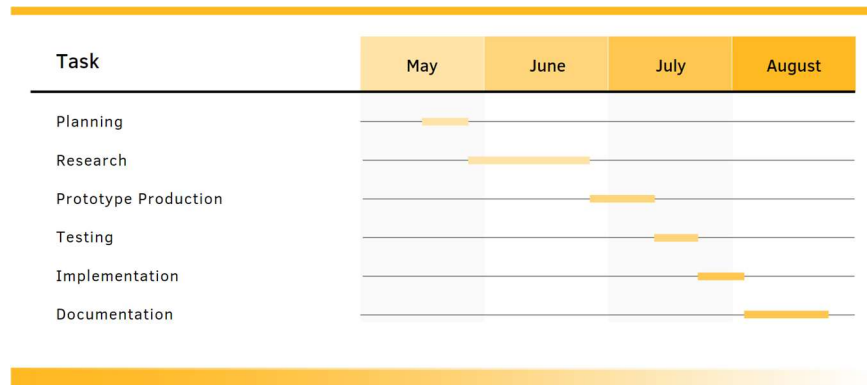
The steps to use the system is simple. First, the user has to select the parameter file by click the Select Parameters button. The text will confirm which parameter file was loaded. Then, the user has to select the images that they wish to predict the crowd count. If they so wish to have a ground truth, they can tick the check box on the left and then, selecting the ground truth file by clicking the Select .CSV file.

After that, the user can click the show density map button to display its estimated density map if needed. Then, the user can predict the set of images by clicking the predict button. After that, the images will be fed into the neural network and the output of the estimated crowd count is displayed into the log below along with the mean inference time as well as the total duration.

To set into video feed mode, the user only has to click the Turn On Video button and it will start estimated the crowd within the video by capturing images every 50ms. To return to image processing mode, the user only has to click the Turn Off Video button.

The system is designed to be used as easy as possible.

Timeline



*Figure 4-7 Gantt Chart of the timeline*

During the middle of May, the planning of the project development had started. An outline of possible tools to use was made and after the planning was made, research was undergone until close to end of June. During this research period, the documentation for PyQt5 was studied and practiced as well. Then, the prototype production began until middle of July where the skeleton of the system was made. After that, testing began, and any bugs found were fixed. Then, the system was implemented using the Mall Dataset images and found to be working great with surveillance style images. Finally, on early of August, the documentation of the project was made.

**CHAPTER 5: SYSTEM EVALUATION AND DISCUSSION**Test Case

<b>Test Case</b>	<b>Test Description</b>	<b>Expected Result</b>	<b>Status</b>
<b>Select Parameter</b>	<ol style="list-style-type: none"> <li>1. Click “Select Parameters” Button</li> <li>2. Select .param file</li> <li>3. Click “Open”</li> </ol>	Selected parameter file will be loaded and shown on GUI	PASS
<b>Predict Images</b>	<ol style="list-style-type: none"> <li>1. Click “Select Parameters” Button</li> <li>2. Select .param file</li> <li>3. Click “Open”</li> <li>4. Click “Select Image(s)”</li> <li>5. Select images</li> <li>6. Click “Open”</li> <li>7. Click “Predict Button”</li> </ol>	Estimated count and mean inference time will be displayed on the log below.	PASS
<b>Predict Images with Ground Truth</b>	<ol style="list-style-type: none"> <li>1. Click “Select Parameters” Button</li> <li>2. Select .param file</li> <li>3. Click “Open”</li> <li>4. Click “Select Image(s)”</li> <li>5. Select images</li> <li>6. Click “Open”</li> <li>7. Tick “Include Ground Truth” checkbox</li> <li>8. Click “Select .CSV file” button</li> <li>9. Select .CSV file</li> <li>10. Click “Select Open”</li> <li>11. Click “Predict Button”</li> </ol>	Estimated and Actual Count and mean inference time will be displayed. A graph between predicted and ground truth will also be displayed.	PASS



<p><b>Predict Images with data saved.</b></p>	<ol style="list-style-type: none"> <li>1. Click “Select Parameters” Button</li> <li>2. Select .param file</li> <li>3. Click “Open”</li> <li>4. Click “Select Image(s)”</li> <li>5. Select images</li> <li>6. Click “Open”</li> <li>7. Tick “Save to .CSV” Button</li> <li>8. Click “Predict Button”</li> </ol>	<p>Estimated count and mean inference time will be displayed on the log below. A folder with the current date and time will be created and inside it will have the .CSV data file.</p>	<p>PASS</p>
<p><b>Predict Video Feed</b></p>	<ol style="list-style-type: none"> <li>1. Click “Select Parameters” Button</li> <li>2. Select .param file</li> <li>3. Click “Open”</li> <li>4. Click “Turn On Video”</li> <li>5. Click “Turn Off Video” after 5 seconds.</li> </ol>	<p>When video is turned on, the displayed video will have the predicted crowd count. Upon turning video off, will return to image processing mode.</p>	<p>PASS</p>
<p><b>Show Density Map</b></p>	<ol style="list-style-type: none"> <li>1. Click “Select Parameters” Button</li> <li>2. Select .param file</li> <li>3. Click “Open”</li> <li>4. Click “Select Image(s)”</li> <li>5. Select images</li> <li>6. Click “Open”</li> <li>7. Click “Show Density Map”</li> </ol>	<p>An estimated density map of the image will be displayed on another window.</p>	<p>PASS</p>

*Table 5-1 Test Plan of the System*

### Discussion

As of the completion of this project, the system does have its up and down. It has the ability to take in only 24-bit images and does not have exception handling if the ground truth file does not have the necessary ground truth for the one of the selected images. Furthermore, the video feed is only taken on the second webcam device of the computer. With that in mind, the system lacks any ability to tweak the settings on the front-end so therefore any settings such as changing the video feed source to the first webcam device will need to be made on the back-end of the system.

What the system can do best however, is able to get input of images and help feed it into the neural network for it to get the prediction count as well as a graph analysis if the ground truth is given. The live video feed feature is also able to get estimated crowd count as the system is able to capture images of the video and send it to the neural network.

## **CHAPTER 6: CONCLUSION**

### Project Review

Upon completion of this project, all the objectives have been met. The system is able to predict images as well as live video feed and also display a graph to compare the prediction and the ground truth if the latter is available. Not only that, but it has also the option to choose the parameter files for the network so that the neural network can be tweaked in the future. The system is also able to traverse the selected images to be viewed.

When studying the documentation for the GUI toolkit, PyQt5, it took me awhile to get to know how the widgets really work and what's their intended use. The object-oriented programming practices within Python is also something that I have learned throughout this project. Regarding the neural network, it is still something that still fascinates me and the community around it is very helpful.

With that said, this system could prove to be useful to users that are keen on learning how neural network works specifically the convolutional neural network and what possibilities it could entail in the future. With the live video feed feature, one can also monitor their surveillance feed to estimate crowd counts around their area.

### Future Work

As mentioned in the previous discussion, there are some future works that could be done with the system. The ability to train and test neural networks will be an interesting addition to the system as it will allow a user to train the neural network from the get-go.

Furthermore, the system will also benefit from a front-end settings tweaking feature so that the users can choose which webcam to use, which neural network model to choose and also what kind of image types to allow based on their liking.

The system can also benefit from multithreading as well as utilize multiple GPUs at the same time which opens up a larger neural network model for the system and an even larger dataset. With multithreading, the system will be able to predict more than one video feed if need be but even then, that feature has to be implemented first.

## **References**

- Brownlee, J., 2019. *Overfitting and Underfitting With Machine Learning Algorithms*. [Online] Available at: <https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/> [Accessed 1 March 2021].
- Chan, A. B., Liang, J. & Vasconcelos, N., 2008. *Privacy Preserving Crowd Monitoring: Counting People without People Models or Tracking*. Anchorage, Alaska, USA, Institute of Electrical and Electronic Engineers, pp. 24-26.
- Dalal, N. & Triggs, B., 2005. *Histograms of Oriented Gradients for Human Detection*. Sand Diego, California, USA, Institute of Electrical and Electronic Engineers.
- Kanwal, S., Khurram, M., Arfeen, M. A. & Li, J., 2018. INTELLIGENT CROWD ANALYSIS BY DETECTING AND COUNTING PEOPLE IN DENSELY CROWDED AERIAL IMAGES. *NED UNIVERSITY JOURNAL OF RESEARCH - THEMATIC ISSUE ON ADVANCES IN IMAGE AND VIDEO PROCESSING*, 15(2), pp. 57-64.
- Kong, D., Gray, D. & Tao, H., 2006. *A Viewpoint Invariant Approach for Crowd Counting*. Hong Kong, China, IEEE.
- Li, M., Zhang, Z., Huang, K. & Tan, T., 2008. *Estimating the number of people in crowded scenes by MID based foreground segmentation and head-shoulder detection*. Tampa, Florida, USA, Institute of Electrical and Electronic Engineers.
- Liu, J., Gao, C., Meng, D. & Hauptmann, A. G., 2018. *DecideNet: Counting Varying Density Crowds Through Attention Guided Detection and Density Estimation*. Salt Lake City, Utah, USA, Institute of Electrical and Electronic Engineers, pp. 5197-5206.
- Miao, Y., Han, J., Gao, Y. & Zhang, B., 2019. ST-CNN: Spatial-Temporal Convolutional Neural Network for crowd counting in videos. *Pattern Recognition Letters*, Volume 125, pp. 113-118.
- Sammut, C. & Webb, G. I., 2011. *Encyclopedia of Machine Learning*. 2010 ed. Boston, MA: Springer.
- Zeiler, M. et al., 2013. On Rectified Linear Units for Speech Processing. *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88, 1988 International Conference on*, pp. 3517-3512.
- Zhang, Y. et al., 2016. *Single-Image Crowd Counting via Multi-Column Convolutional Neural Network*. Las Vegas, Institute of Electrical and Electronics Engineers.

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: T3, Y3</b>	<b>Study week no.: 2</b>
<b>Student Name &amp; ID: Mohd Wafi Nazrul Adam, 17ACB05930</b>	
<b>Supervisor: Mr Teoh Shen Khang</b>	
<b>Project Title: Visual Crowd Counting System using Deep Learning</b>	

## 1. WORK DONE

Browsing the toolkits to use for the development of the system.

Training neural network with dataset.

## 2. WORK TO BE DONE

Browse for further information regarding toolkits for development.


## 3. PROBLEMS ENCOUNTERED

None so far.

## 4. SELF EVALUATION OF THE PROGRESS

Only steppingstone so far. More work still needs to be done.

  
Supervisor's signature

  
Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: T3, Y3</b>	<b>Study week no.: 4</b>
<b>Student Name &amp; ID: Mohd Wafi Nazrul Adam, 17ACB05930</b>	
<b>Supervisor: Mr Teoh Shen Khang</b>	
<b>Project Title: Visual Crowd Counting System using Deep Learning</b>	

## 1. WORK DONE

Chose the toolkit for GUI development of the system, PyQt5

Reading documentation of the GUI and tutorials online.

## 2. WORK TO BE DONE

Develop sketch GUI and plan system capabilities.

Neural network needs further fine tuning. On hold for GUI development for now.

## 3. PROBLEMS ENCOUNTERED

Neural Network giving NaN output. Might have something to do with dataset input.

## 4. SELF EVALUATION OF THE PROGRESS

Need more time on reading documentation.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: T3, Y3</b>	<b>Study week no.: 6</b>
<b>Student Name &amp; ID: Mohd Wafi Nazrul Adam, 17ACB05930</b>	
<b>Supervisor: Mr Teoh Shen Khang</b>	
<b>Project Title: Visual Crowd Counting System using Deep Learning</b>	

## 1. WORK DONE

Developed sketch of the GUI and it's first and foremost capabilities.

Neural network finally gives accurate result on Test dataset.

## 2. WORK TO BE DONE

Connect GUI with Neural Network through functions.


Set up Dataloader to work with GUI as well.


## 3. PROBLEMS ENCOUNTERED

Documentation of the toolkit is messy. Too much deprecated methods still listed inside.

## 4. SELF EVALUATION OF THE PROGRESS

Still need more time with GUI documentation.

  
Supervisor's signature

  
Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: T3, Y3</b>	<b>Study week no.: 8</b>
<b>Student Name &amp; ID: Mohd Wafi Nazrul Adam, 17ACB05930</b>	
<b>Supervisor: Mr Teoh Shen Khang</b>	
<b>Project Title: Visual Crowd Counting System using Deep Learning</b>	

## 1. WORK DONE

System can finally send input images to neural network but buggy at the moment.

Dataloader can somewhat work with the input images.

## 2. WORK TO BE DONE

Fix bugs regarding input images

Add additional functionalities.


## 3. PROBLEMS ENCOUNTERED

Input images giving wrong tensor channels into neural network causing errors.

## 4. SELF EVALUATION OF THE PROGRESS

More work needs to be done.

  
Supervisor's signature

  
Student's signature



# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: T3, Y3</b>	<b>Study week no.: 10</b>
<b>Student Name &amp; ID: Mohd Wafi Nazrul Adam, 17ACB05930</b>	
<b>Supervisor: Mr Teoh Shen Khang</b>	
<b>Project Title: Visual Crowd Counting System using Deep Learning</b>	

## 1. WORK DONE

System can finally work with input images but only with 24-bit color.

System can display density map of the image.

## 2. WORK TO BE DONE


Add additional functionalities such as ground truth comparison.

## 3. PROBLEMS ENCOUNTERED

Re-selecting image files will cause errors of index of arrays being out of bounds.

## 4. SELF EVALUATION OF THE PROGRESS

Bugs need to be fixed as soon as possible and maybe try adding more functionalities.

  
Supervisor's signature

  
Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: T3, Y3</b>	<b>Study week no.: 12</b>
<b>Student Name &amp; ID: Mohd Wafi Nazrul Adam, 17ACB05930</b>	
<b>Supervisor: Mr Teoh Shen Khang</b>	
<b>Project Title: Visual Crowd Counting System using Deep Learning</b>	

## 1. WORK DONE

System can change parameter files, estimate images from set of image files or video feed.

Fixed some known bugs, developed test plan for system as well.

Minor UI update to system.

## 2. WORK TO BE DONE

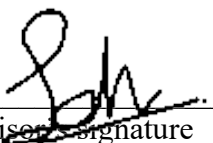
Report needs to be done.

## 3. PROBLEMS ENCOUNTERED

Functionalities giving errors once in a while but already fixed.

## 4. SELF EVALUATION OF THE PROGRESS

System is completed.

  
Supervisor's signature



Student's signature

# Visual Crowd Counting System Using Deep Learning

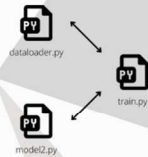
Name: Mohd Wafi Nazrul Adam  
ID: 1705930  
Supervisor: Mr Teoh Shen Khang

Faculty of Information and Computer Technology  
UTAR  
Computer Engineering | Final Year Project

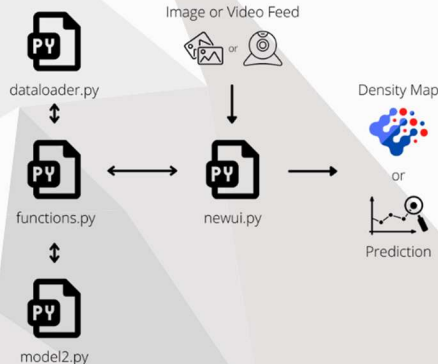
## Introduction:

A system that visually counts the crowd using Deep Learning through input of images or video feed with the help of a GUI.

## Neural Network Training:



## System Diagram:



## Method:

Neural network will need to be trained prior.

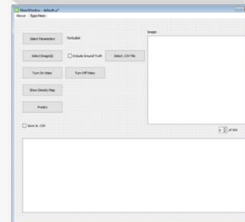
Mall Dataset will be used for training.

Develop the GUI using PyQt5 toolkit.

System will be able to take in image or video feed where it will be sent to the neural network to get processed and predict the crowd count.

System will be able to change the parameter of the neural network model for better versatility.

## GUI Sketch:



## Discussion:

Neural network is based on MCNN but with an additional layer of Batch Normalization on each of the convolutional neural network layer.

Trained neural network has MAE of 2.45 and MSE of 9.72.

System able to take in video feed and capture image every 50ms to predict crowd count and display data in log.

System is also able to take in images with ground truth to measure the accuracy of the network if needed. A graph will be displayed.

## Conclusion:

System development was successful.

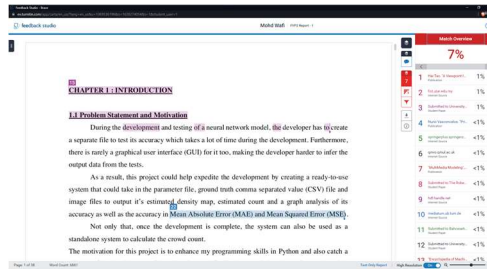
Object-oriented practices programming helped expedite the development.

## Possible Future Work:

Add settings tweak on front-end

Add ability to swap neural network model

# Plagiarism Check Result



## Turnitin Originality Report

FYP2 Report - 1 by Mohd Wafi  
From FYP2 report (FYP2 report)

Processed on 30-Aug-2021 22:37 +08  
ID: 1638274054  
Word Count: 6861

Similarity Index	Similarity by Source
7%	Internet Sources: 4%
	Publications: 5%
	Student Papers: 4%

### sources:

- 1 1% match (publications)  
[Hai Tao, "A Viewpoint Invariant Approach for Crowd Counting", 18th International Conference on Pattern Recognition \(ICPR 06\), 2006](#)
- 2 1% match (Internet from 18-Jul-2021)  
[https://fict.utar.edu.my/documents/FYP/FYP\\_guidelines/FYP2\\_Guidelines.docx](https://fict.utar.edu.my/documents/FYP/FYP_guidelines/FYP2_Guidelines.docx)
- 3 1% match (student papers from 10-Jun-2020)  
[Submitted to University of Leicester on 2020-06-10](#)
- 4 < 1% match (publications)  
[Nuno Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking", 2008 IEEE Conference on Computer Vision and Pattern Recognition, 06/2008](#)
- 5 < 1% match (Internet from 02-May-2021)  
<https://springerplus.springeropen.com/articles/10.1186/s40064-015-1427-3>
- 6 < 1% match ()  
[Chen, Ke, "Latent Dependency Mining for Solving Regression Problems in Computer Vision", "Queen Mary University of London", 2013](#)
- 7 < 1% match (publications)  
["MultiMedia Modeling", Springer Science and Business Media LLC, 2018](#)
- 8 < 1% match (student papers from 04-Sep-2014)  
[Submitted to The Robert Gordon University on 2014-09-04](#)
- 9 < 1% match ()  
[Zhang Q., Tolia G., Mansencal B., Saracoglu A. et al. "COST292 experimental framework for TRECVID 2008", NIST, 2008](#)
- 10 < 1% match ()



**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

<b>Full Name(s) of Candidate(s)</b>	Mohd Wafi Nazrul Adam Bin Mohd Ridhwan Oxley Adam
<b>ID Number(s)</b>	1705930
<b>Programme / Course</b>	CT
<b>Title of Final Year Project</b>	Visual Crowd Counting System using Deep Learning

<b>Similarity</b>	<b>Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)</b>
<b>Overall similarity index: <u>7</u> %</b>  <b>Similarity by source</b> Internet Sources: <u>4</u> % Publications: <u>5</u> % Student Papers: <u>4</u> %	
<b>Number of individual sources listed of more than 3% similarity: <u>0</u></b>	
<b>Parameters of originality required and limits approved by UTAR are as Follows:</b> (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

*Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.*

  
 \_\_\_\_\_  
 Signature of Supervisor  
 Name: Teoh Shen Khang  
 Date: 3 September 2021

\_\_\_\_\_  
 Signature of Co-Supervisor  
 Name: \_\_\_\_\_  
 Date: \_\_\_\_\_



## UNIVERSITI TUNKU ABDUL RAHMAN



### FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

#### CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	17ACB05930
Student Name	Mohd Wafi Nazrul Adam Bin Mohd Ridhwan Oxley Adam
Supervisor Name	Teoh Shen Khang

TICK (✓)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
✓	Front Cover
✓	Signed Report Status Declaration Form
✓	Title Page
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	References
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

\*Include this form (checklist) in the thesis (Bind together as the last page)

<p>I, the author, have checked and confirmed all the items listed in the table are included in my report.</p> <div style="text-align: center;">   <hr style="width: 100px; margin: 0 auto;"/>         (Signature of Student)          Date: 30/08/2021       </div>	<p>Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.</p> <div style="text-align: center;">   <hr style="width: 100px; margin: 0 auto;"/>         (Signature of Supervisor)          Date: 3 September 2021       </div>
--	---