CT104-3-M-PR
PATTERN RECOGNITION

ASSIGNMENT 2

Instruction:

- Marks will be awarded for good report and thoroughness in your approach.
- Referencing Code: If you use some code, or ideas for code, which are taken or adapted from another source (book, magazine, internet, discussion forum, etc), then this **must** be cited and referenced using the Harvard Name convention within your source code. Failure to reference code properly is considered as plagiarism.
- Complete this cover sheet and attach it to your project.
- This project is to be attempted by an individual student.

Student declaration:

I declare that:
- *I understand what is meant by plagiarism*
- *The implication of plagiarism has been explained to me by our lecturer*
- *This project is all my work and I have acknowledged any use of the published or unpublished works of other people.*

Student Signature: ***Kyle Lai*** .... Date: 27th DECEMBER 2020 ....

Intake:	APUMF2006AI(PR)
Project Title:	**Crack Detection in Concrete Surface for Building Safety Inspection Part 2**
Name	Lai Kai Lok
Signature	***Kyle Lai***
TP Number	TP061241
Lecturer	Dr. Chandra Reka A/P Ramachandiran

TABLE OF CONTENTS

TABLE OF CONTENTS ii
LIST OF TABLES iii
LIST OF FIGURES iv
LIST OF ABBREVIATIONS v
FORMULATION AND DESIGN 1
1.1 Classification Stage 1
1.1.1 GoogLeNet 1
1.2 Crack Area Localization and Extraction Stage 3
1.2.1 Object detection 3
1.2.2 Semantic segmentation 4
1.3 Design of GUI 5
IMPLEMENTATION 8
2.1 Data Preparation Phase 8
2.2 Model Development Phase 8
2.2.1 Classification Model 8
2.2.2 Localization and Extraction Models 10
2.2.2.1 YOLO v2 Object Detection Model 11
2.2.2.2 Unet and ResNet-18 Semantic Segmentation Model 15
2.2.2.3 Localization and Extraction Models Evaluation 18
2.3 GUI Building Phase 21
RESULTS 25
3.1 System Evaluation 25
3.1.1 Examples of System Testing 25
3.1.2 Performance of Classification and Region of Interest (ROI) Extraction 32
3.2 Strengths and Limitations 33
CONCLUSIONS 35
REFERENCES 36
APPENDICE 37

LIST OF TABLES

Table 1.1: The architecture of GooLeNet (Szegedy *et al.*, 2015). 1
Table 1.2: The architecture of Unet (Liu *et al.*, 2019) 4
Table 2.1: The performance of the 3 ROI Extraction Models 21
Table 3.1: Screenshots for the 20 examples of system testing 25
Table 3.2: Performance of the system in classification and ROI extraction 32

LIST OF FIGURES

Figure 1.1: The flow for the development of crack detection system 2

Figure 1.2: Road crack detection by YOLO v2 (Mandal, Uong and Adu-Gyamfi, 2019) 3

Figure 1.3: Crack detection by using object detection, above and semantic segmentation, below (Zhang *et al.*, 2019). 5

Figure 1.4: The design of the GUI 5

Figure 1.5: The working flow of the GUI system. 7

Figure 2.1: The training of classification model wallnet 10

Figure 2.2: Confusion matrix of test dataset using model wallnet for classification. 10

Figure 2.3: The ground truth box labeling process 11

Figure 2.4: The aspect ratio and area of boxes in the Gtruth matrix 13

Figure 2.5: The optimal number of anchor box 13

Figure 2.6: The training of object detection model YOLO v2 15

Figure 2.7: The segmentation region labeling process 16

Figure 2.8: The training of segmentation model unet 17

Figure 2.9: The training of segmentation model ResNet-18 17

Figure 2.10: The performance of ROI extraction models in test set 01 18

Figure 2.11: The performance of ROI extraction models in test set 02 19

Figure 2.12: The performance of ROI extraction models in test set 03 19

Figure 2.13: The performance of ROI extraction models in test set 04 19

Figure 2.14: The performance of ROI extraction models in test set 05 19

Figure 2.15: The performance of ROI extraction models in test set 06 20

Figure 2.16: The performance of ROI extraction models in test set 07 20

Figure 2.17: The performance of ROI extraction models in test set 08 20

Figure 2.18: The performance of ROI extraction models in test set 09 20

Figure 2.19: The performance of ROI extraction models in test set 10 21

Figure 2.20: The process of GUI designing 22

Figure 2.21: GUI when no crack is detected in the image. 24

Figure 2.22: GUI when no crack is detected in the image. 24

Figure 3.1: System testing on random image obtained from google image seach 33

Figure 3.2: System testing on random image that does not contain crack 33

LIST OF ABBREVIATIONS

CNN.................... Convolutional Neural Network

YOLO................ You Only Look Once

ResNet .... Residual Nets

GUI..................... Graphical User Interface

TP .... True Positives

FP........................ False Positives

TN....................... True Negatives

FN....................... False Negatives

ROI .... Region of Interest

SECTION 1

FORMULATION AND DESIGN

The crack detection system is divided into 2 stage processes. The first stage is the classification stage, which categorizes the input image into a crack or non-crack group. Only those images which were categorized as crack will proceed to the next stage for crack area localization and extraction. The first stage is developed by using GoogLeNet deep learning framework. Comparison between object detection and semantic segmentation are made in the second stage. Figure 1.1 illustrated the flow and design of the crack detection system.

1.1 Classification Stage

Concrete images collected are used to train the GoogLeNet so that it can differentiate between a 'Crack' image and a 'Non-Crack' image.

1.1.1 GoogLeNet

GoogLeNet is a type of deep convolutional neural network (CNN) which has a depth of 22 layers and 7 million parameters (Szegedy *et al.*, 2015). This network is mainly used for classification and detection. Its main advantages is huge performance improvement with moderate rise of computational power compared with smaller network architecture.

Table 1.1: The architecture of GooLeNet (Szegedy *et al.*, 2015).

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

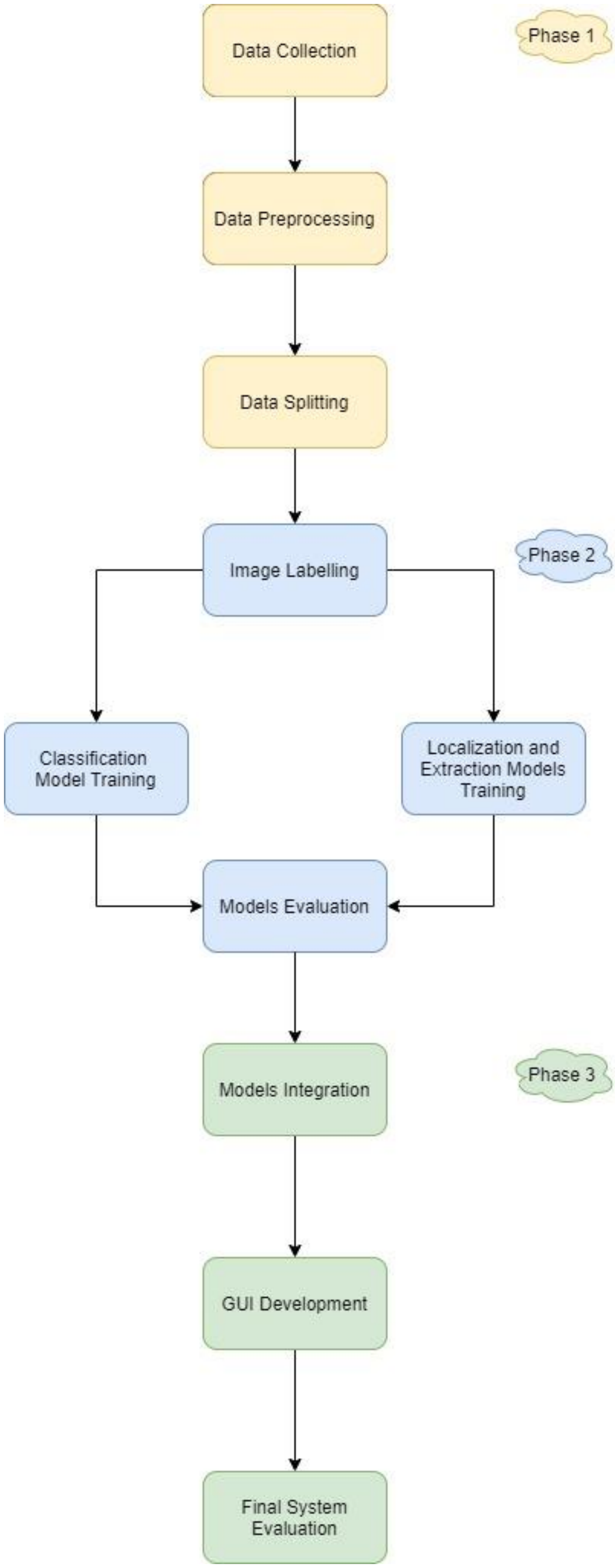


Figure 1.1: The flow for the development of crack detection system

1.2 Crack Area Localization and Extraction Stage

The performance of object detection technique and semantic segmentation technique for localizing and extracting the crack area are compared.

1.2.1 Object detection

Object detection technique uses different size of bounding boxes or rectangular boxes to scan through the image and check whether the object of interest is in each box or not (Liu *et al.*, 2019). This method mainly detects and localizes the object by using its aspect ratio. In this study, YOLO v2 will be applied for locating crack regions in concrete images.

YOLO v2 is a fast and accurate object detection system which can detect different objects while balancing the accuracy performance and the computational time needed (Redmon and Farhadi, 2017). A YOLO v2 framework was trained for 18.2 hours by using 9053 road images and transfer learning techniques for detecting crack area on the road images. The model does not perform well in detecting transverse types of cracking area (Mandal, Uong and Adu-Gyamfi, 2019).

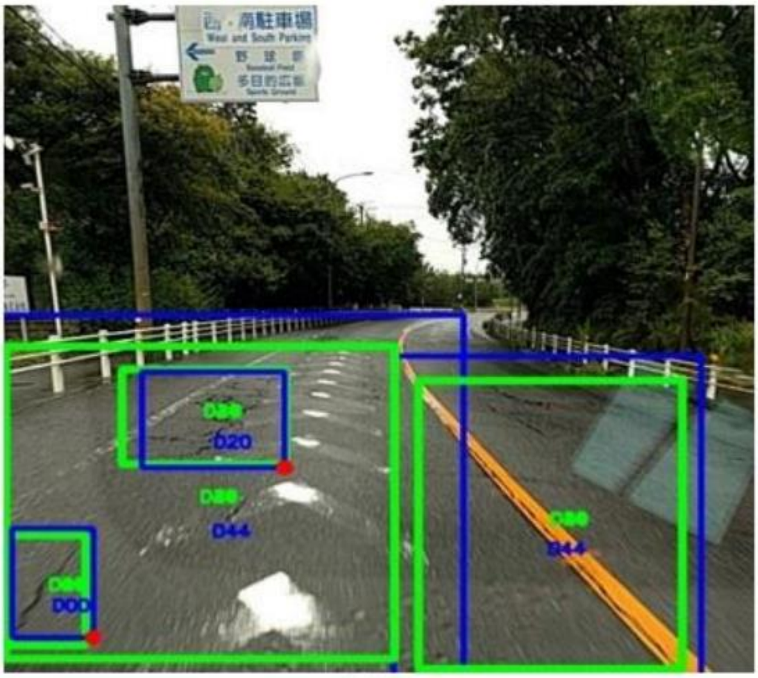


Figure 1.2: Road crack detection by YOLO v2 (Mandal, Uong and Adu-Gyamfi, 2019)

1.2.2 Semantic segmentation

Semantic segmentation techniques conduct the region division task by classifying each pixel in the image into its respective category (Liu *et al.*, 2019). In this study, the Unet and residual nets (ResNet-18) are used as the base framework for concrete crack semantic segmentation and their performance is compared.

Unet is mainly used for biomedical image segmentation with encoder-decoder structures. Features were extracted in the encoder and pass to decoder for fine features repairing to improve the precision. Unet was first used for crack detection in the study of (Liu *et al.*, 2019), which showed better performance than other CNN architecture and object detection techniques.

Table 1.2: The architecture of Unet (Liu *et al.*, 2019)

Layers		Image size	Operation	Image size	Convolution kernel	Step size	Edge filling
IN	Input	$3 \times 512 \times 512$	Conv1 + BN + ReLU	$3 \times 3 \times 3$	64	1	1
CB1	L1	$64 \times 512 \times 512$	Conv2 + BN + ReLU	$64 \times 3 \times 3$	64	1	1
	L2	$64 \times 512 \times 512$	MaxPooling	2×2	–	2	–
CB2	L3	$64 \times 256 \times 256$	Conv3 + BN + ReLU	$64 \times 3 \times 3$	128	1	1
	L4	$128 \times 256 \times 256$	Conv4 + BN + ReLU	$128 \times 3 \times 3$	128	1	1
	L5	$128 \times 256 \times 256$	MaxPooling	2×2	–	2	–
CB3	L6	$128 \times 128 \times 128$	Conv5 + BN + ReLU	$128 \times 3 \times 3$	256	1	1
	L7	$256 \times 128 \times 128$	Conv6 + BN + ReLU	$256 \times 3 \times 3$	256	1	1
	L8	$256 \times 128 \times 128$	MaxPooling	2×2	–	2	–
CB4	L9	$256 \times 64 \times 64$	Conv7 + BN + ReLU	$256 \times 3 \times 3$	512	1	1
	L10	$512 \times 64 \times 64$	Conv8 + BN + ReLU	$512 \times 3 \times 3$	512	1	1
	L11	$512 \times 64 \times 64$	MaxPooling	2×2	–	2	–
CB5	L12	$512 \times 32 \times 32$	Conv9 + BN + ReLU	$512 \times 3 \times 3$	1024	1	1
	L13	$1024 \times 32 \times 32$	Conv10 + BN + ReLU	$1024 \times 3 \times 3$	1024	1	1
	L14	$1024 \times 32 \times 32$	ConvTrans1	$1024 \times 2 \times 2$	512	2	–
CB6	L15	$512 \times 64 \times 64$	Cat L11	–	–	–	–
	L15 + L11	$1024 \times 64 \times 64$	Conv11 + BN + ReLU	$1024 \times 3 \times 3$	512	1	1
	L16	$512 \times 64 \times 64$	Conv12 + BN + ReLU	$512 \times 3 \times 3$	512	1	1
	L17	$512 \times 64 \times 64$	ConvTrans2	$512 \times 2 \times 2$	256	2	–
CB7	L18	$256 \times 128 \times 128$	Cat L8	–	–	–	–
	L18 + L8	$512 \times 128 \times 128$	Conv13 + BN + ReLU	$512 \times 3 \times 3$	256	1	1
	L19	$256 \times 128 \times 128$	Conv14 + BN + ReLU	$256 \times 3 \times 3$	256	1	1
	L20	$256 \times 128 \times 128$	ConvTrans3	$256 \times 2 \times 2$	128	2	–
CB8	L21	$128 \times 256 \times 256$	Cat L5	–	–	–	–
	L21 + L5	$256 \times 256 \times 256$	Conv15 + BN + ReLU	$256 \times 3 \times 3$	128	1	1
	L22	$128 \times 256 \times 256$	Conv16 + BN + ReLU	$128 \times 3 \times 3$	128	1	1
	L23	$128 \times 256 \times 256$	ConvTrans4	$128 \times 2 \times 2$	64	2	–
CB9	L24	$64 \times 512 \times 512$	Cat L2	–	–	–	–
	L24 + L2	$128 \times 512 \times 512$	Conv17 + BN + ReLU	$128 \times 3 \times 3$	64	1	1
	L25	$64 \times 512 \times 512$	Conv18 + BN + ReLU	$64 \times 3 \times 3$	128	1	1
	L26	$64 \times 512 \times 512$	Conv19 + Softmax	$64 \times 1 \times 1$	1	1	0
OUT	output	$2 \times 512 \times 512$					

ResNet-18 is a CNN with 18 layers of depth and 11.7 million parameters. ResNet can perform well in extracting features (Lopes and Valiati, 2017). ResNet-18 was chosen for crack detection as it required less computational time with satisfied accuracy performance (Zhang *et al.*, 2019).

All model training is done by using transfer learning techniques in the MATLAB platform. These training are conducted by using a single Intel core i5-4200, 1.6 GHz CPU.

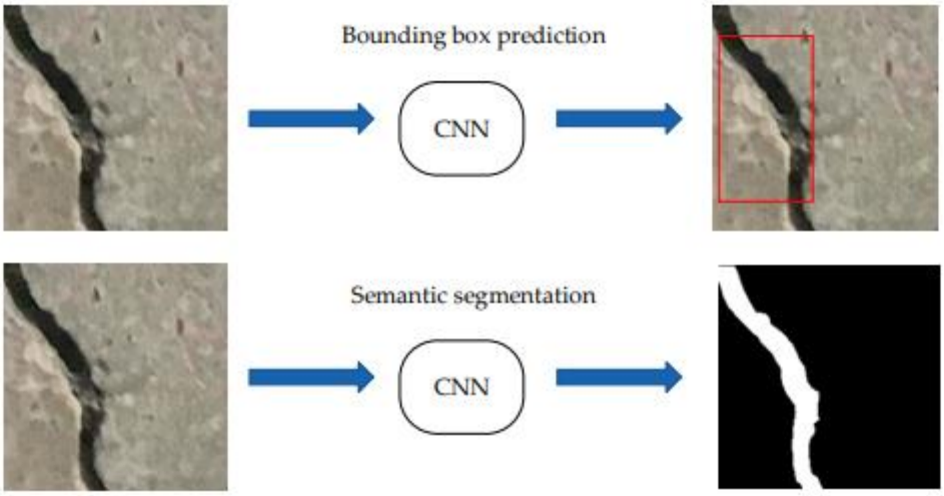


Figure 1.3: Crack detection by using object detection, above and semantic segmentation, below (Zhang *et al.*, 2019).

1.3 Design of GUI

The GUI of the crack detection system is developed by using the MATLAB Design App. The application is named as "CrackTect System".

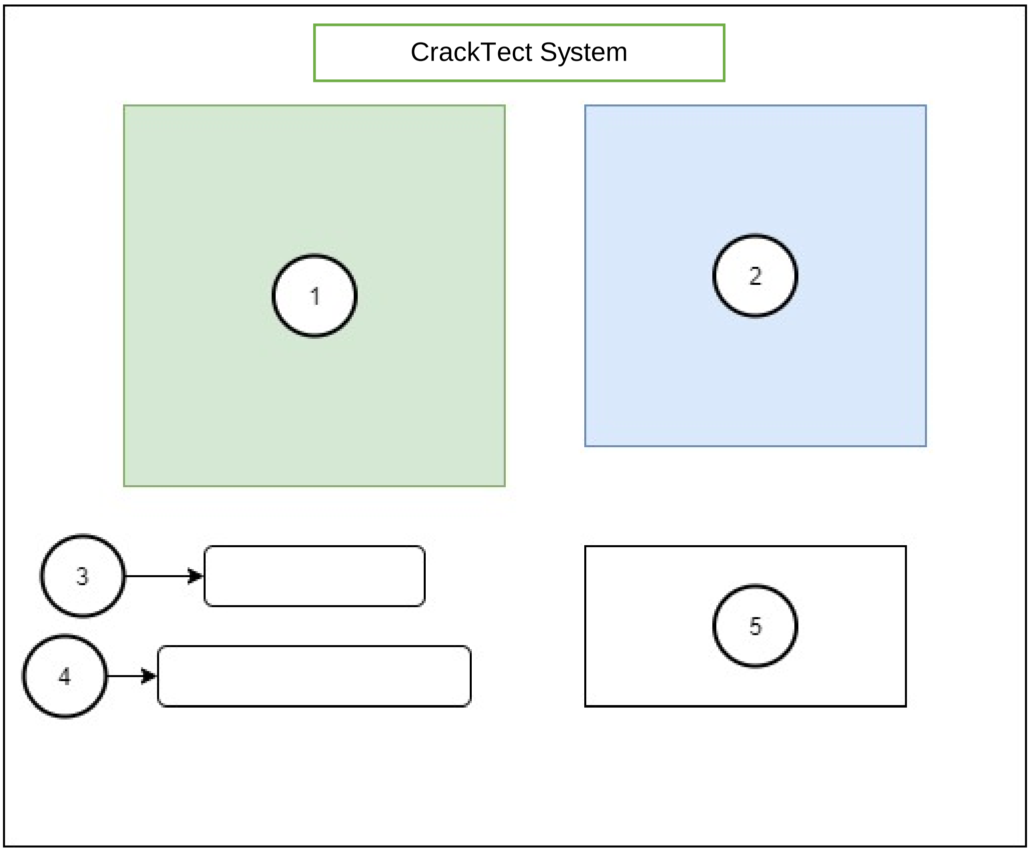


Figure 1.4: The design of the GUI

The draft of the system GUI is illustrated in Figure 1.4. There are 5 main component of in the GUI, which are:

1. The original picture frame: displaying the selected original image before analysed by the system.

2. The result picture frame: if there is no crack detected in the picture, this area will display the original image. If a crack is detected, this area will be displaying an image where the crack area is highlighted, either in a bounding box or using a different colour representing the crack area, depending on the model being used, either the object detection model or the semantic segmentation model.

3. “Random Image” button: by clicking this button, the system will randomly select one of the images from the collected dataset to be analysed.

4. “Load Custom Image” button: by clicking this button, the users can select the desired image to be analysed.

5. Crack detection message box: if there is no crack detected in the picture, this area will display “There is no Crack”. If a crack is detected, the message box will display the message “Crack is detected. Crack Area is highlighted with Blue colour” or “Crack is detected. The Crack Area is highlighted with a Yellow box” depending on the models being used in the localization and extraction stage.

To exit the app, users will have to click the cross button located in the right top corner in the GUI. The working flow of the GUI system is shown in Figure 1.5.

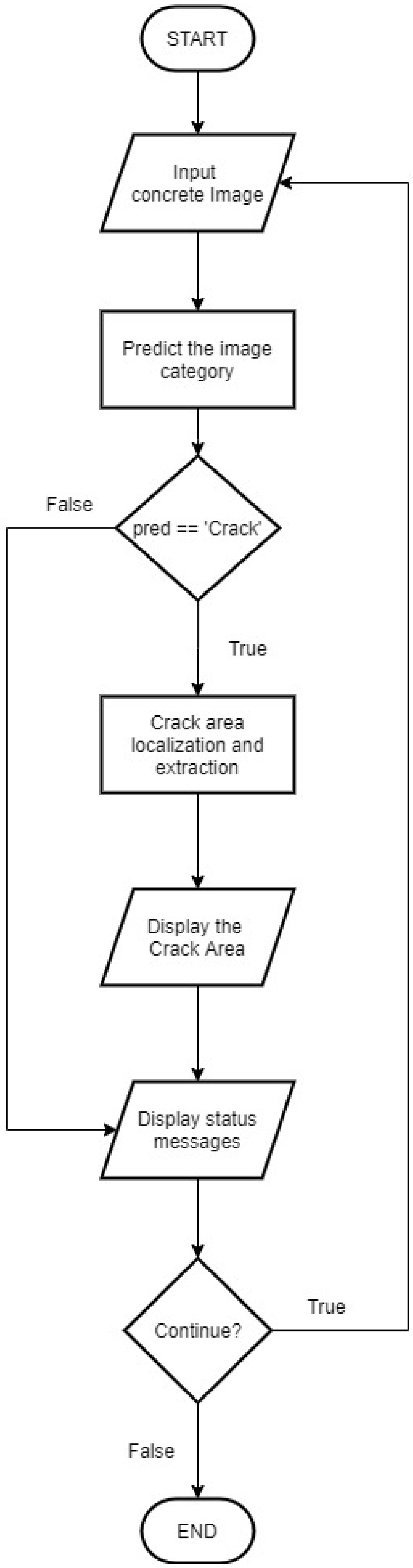


Figure 1.5: The working flow of the GUI system.

SECTION 2

IMPLEMENTATION

Three phases are involved in developing the crack detection system. These phases are the data preparation phase, the model development phase and the graphical user interface (GUI) building phase.

2.1 Data Preparation Phase

All the concrete images used in this study are obtained from Mendeley website (Çağlar Fırat Özgenel, 2019). The dataset contains 40,000 227 × 227 pixels RGB concrete images, where 20,000 of the images are having cracks while the other 20,000 are without cracks. Due to the limitation of the computational power, only 300 images from each class are used in this study. All the 600 images are resized into 224 × 224 pixels before model training to fulfil the input size requirement of GoogLeNet and ResNet-18.

To reduce the amount of computational time for model training, different numbers of train data are used for different types of model training and check if less training data is able to create an effective system or not. Generally, 120 images from each class are set as training dataset. The exact number of images used to train each model is presented in the model development phase.

2.2 Model Development Phase

One model is developed for the classification stage. The model is developed based on GoogLeNet. For the crack area localization and extraction, 3 different models are trained and compared. The best model is selected for the final system development.

2.2.1 Classification Model

120 crack images are placed in a folder named as "Positive" and another 120 normal non-crack images are placed in a folder named as "Negative". Both folders are then placed in the same folder named as "Crack_detection_samples".

```
wallds = imageDatastore("Crack_detection_samples","IncludeSubfolders",...
    true,"LabelSource","foldernames");
```

The code above loads all the images from the Crack_detection_sample folder into MATLAB and labels each image with the subfolder's name it belongs to.

```
[trainImgs,testImgs] = splitEachLabel(wallds,0.6);

numClasses = numel(categories(wallds.Labels));
```

The code above split the dataset into train images and validation images in the ratio of 0.6:0.4. The number of classes is then set as 2, following the number of label types in the dataset.

```
11  net = googlenet;
12  lgraph = layerGraph(net);
13
14  newFc = fullyConnectedLayer(2,"Name","new_fc");
15  lgraph = replaceLayer(lgraph,"loss3-classifier",newFc);
16  newOut = classificationLayer("Name","new_out");
17  lgraph = replaceLayer(lgraph,"output",newOut);
18
19
20  options = trainingOptions("sgdm","InitialLearnRate", 0.001);
21
22  testLabels = testImgs.Labels;
23  inputSize=[224 224];
24  trainImgs = augmentedImageDatastore(inputSize, trainImgs);
25  testImgs = augmentedImageDatastore(inputSize, testImgs);
26
27  [wallnet,info] = trainNetwork(trainImgs, lgraph, options);
```

The code above first set the GoogLeNet as the backbone network. Then the last hidden layer is replaced with 2 fully connected node layer and the output layer is replaced with a 2 class classification layer. After that, the training option is set to use the sgdm method and the learning rate is set as 0.001. The size of all the images are then resize to 224 × 224 to fit the requirement of GoogLeNet. The line 27 code start the training of the new network and named it as "wallnet". The whole process is called transfer learning as it only change few layers of the existing network for specific purpose, in this study, it is to differentiate between a crack image and a non-crack image.

The training of wallnet takes about 1 hour and 45 minutes to complete. It is able to achieve 100% accuracy for both the train and validation dataset. The training details of wallnet is shown in Figure 2.1. The model is then save as model1 so that it can be deployed later for system development. The trained model is then evaluated by using 360 test images and able to achieve an accuracy rate of 99.17%.

```
>> crack_detection
Training on single CPU.
Initializing input data normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:03:59 |       42.97% |       1.1335 |          0.0010 |
|      30 |          30 |       01:40:54 |      100.00% |       0.0005 |          0.0010 |
|========================================================================================|
     1
```

Figure 2.1: The training of classification model wallnet

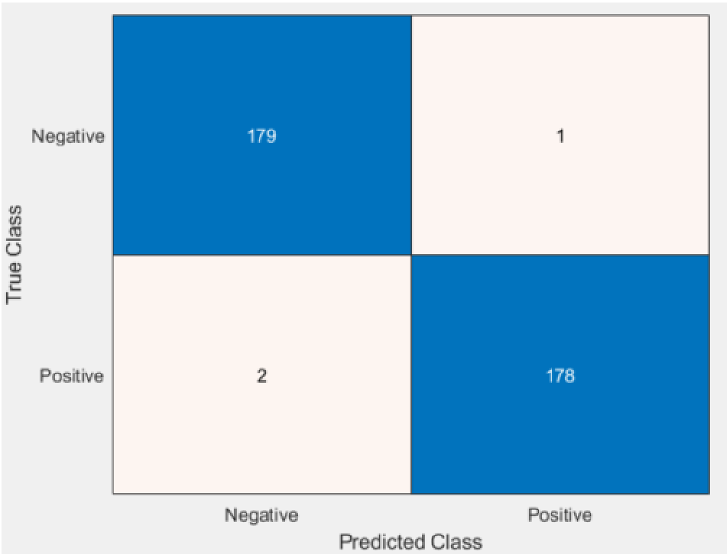


Figure 2.2: Confusion matrix of test dataset using model wallnet for classification.

2.2.2 Localization and Extraction Models

In this section, the object detection method and the segmentation methods are being implemented and compared. As the model main purpose is to localize and extract the crack area, only crack images are used for training.

2.2.2.1 YOLO v2 Object Detection Model

120 crack images are labelled with rectangular box to highlight the crack area by using the image labeller application in MATLAB. The labelled dataset are then save as Gtruth matrix so that it can be used to train the YOLO v2 model later.

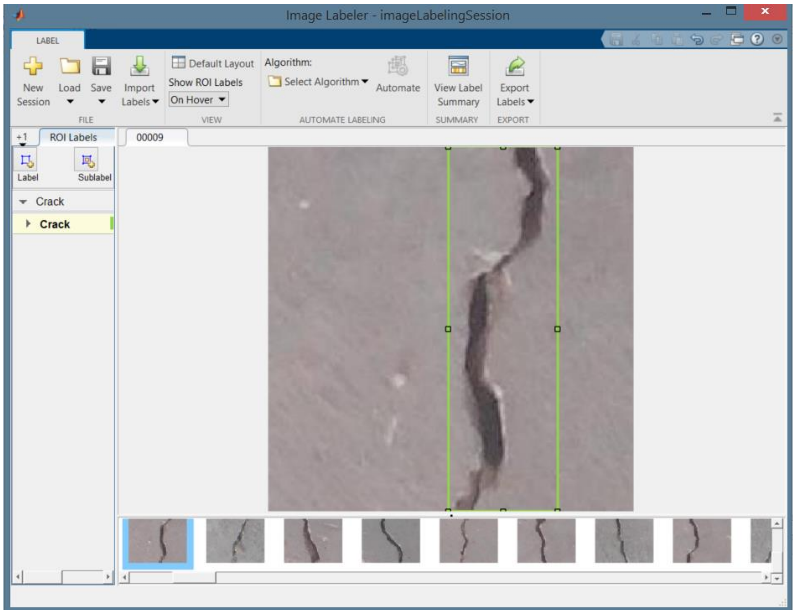


Figure 2.3: The ground truth box labeling process

The number of anchor boxes need to be defined while training the YOLO v2 model. The ground truth labelled data from the previous section can be used to estimate the optimal number of anchor boxes.

```
data = load('Gtruth.mat');
crackDataset = data.gTruth.LabelData;
data.gTruth.DataSource.Source;
crackDataset(1:4,:)

allBoxes = vertcat(crackDataset.Crack{:});
aspectRatio = allBoxes(:,3) ./ allBoxes(:,4);
area = prod(allBoxes(:,3:4),2);

figure
scatter(area,aspectRatio)
xlabel("Box Area")
ylabel("Aspect Ratio (width/height)");
title("Box Area vs. Aspect Ratio")

trainingData = boxLabelDatastore(crackDataset(:,:));

numAnchors = 5;
[anchorBoxes,meanIoU] = estimateAnchorBoxes(trainingData,numAnchors);

maxNumAnchors = 15;
meanIoU = zeros([maxNumAnchors,1]);
anchorBoxes = cell(maxNumAnchors, 1);
for k = 1:maxNumAnchors
    % Estimate anchors and mean IoU.
    [anchorBoxes{k},meanIoU(k)] = estimateAnchorBoxes(trainingData,k);
end

figure
plot(1:maxNumAnchors,meanIoU,'-o')
ylabel("Mean IoU")
xlabel("Number of Anchors")
title("Number of Anchors vs. Mean IoU")
```

The code above estimates the optimal number of anchor boxes from the Gtruth matrix data. Figure 2.3 showed that the aspect ratio of the bounding box highlighting the crack area does not have a consistent value, it varies from 0 to 6. This is understandable as the shape of the crack area is never consistent. Thus, the object detection model is likely not able to perform well in localizing and extracting the crack area. Figure 2.3 showed that 10 is the optimal number of anchor boxes to be used for YOLO v2 model training.

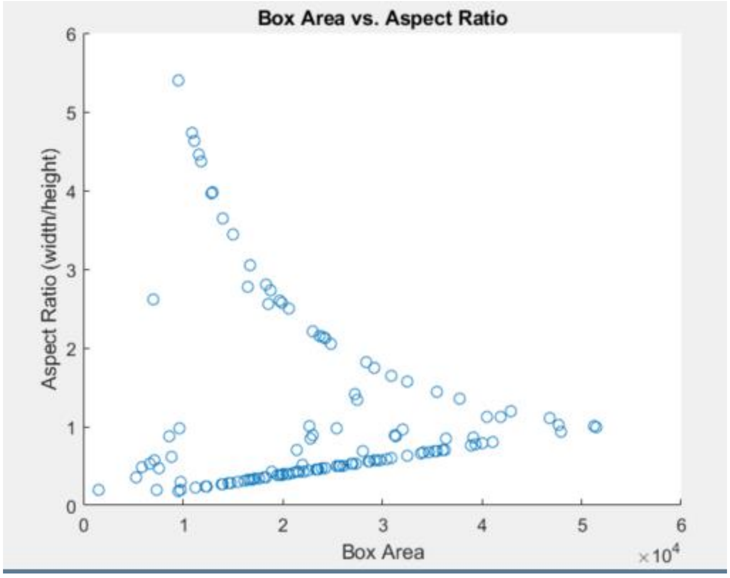


Figure 2.4: The aspect ratio and area of boxes in the Gtruth matrix

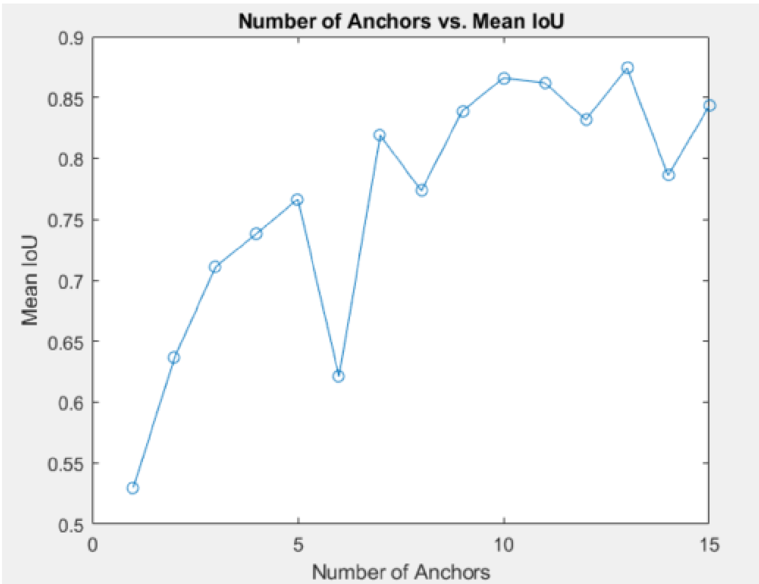


Figure 2.5: The optimal number of anchor box

The next step is to train a YOLO v2 model.

```
data = load('Gtruth.mat');
lblBox = boxLabelDatastore(data.gTruth.LabelData);
imPath = imageDatastore(data.gTruth.DataSource.Source);
crackData = combine(imPath, lblBox);
scaledData = transform(crackData,@scaleGT);

anchorBoxes = estimateAnchorBoxes(scaledData,10);
```

The code above combines the ground truth box and the crack images, then it resizes the combined data into the required size to train a ResNet-18, which is the backbone to be used for building the YOLO v2 model. The line 7 code sets the number of anchor boxes as 10, which was obtained from the previous section.

```
19 -    net = resnet18;
20 -    numClasses = 1;
21 -    imageSize = [224 224 3];
22
23 -    lgraph = yolov2Layers(imageSize,numClasses,anchorBoxes,net,...
24           "res5b_relu","ReorgLayerSource","res3a_relu");
25
26 -    options = trainingOptions('sgdm', ...
27               'MiniBatchSize',16, ....
28               'InitialLearnRate',1e-3, ...
29               'MaxEpochs',20,...
30               'VerboseFrequency',10);
31
32 -    detector = trainYOLOv2ObjectDetector(scaledData,lgraph,options);
```

The code above sets ResNet-18 as the backbone network for the YOLO v2 model, fixes the number of classes as 1 which is to detect the crack area and sets the required image input size. Line 24 set the feature extraction layer to res5b_relu and the res3a_relu as the reorganization layer. This is set for transfer learning so that training of model does not need to start from scratch. The option sets sgdm as the training method, with mini batch size as 16, learning rate as 0.001, maximum epochs as 20 and the frequency of printing the progress of the training is set as 10. Line 32 starts the training and saved the trained model as detector. The trained model is saved as the yolov2Model2 so that it can be deployed later for system development.

```
*************************************************************************
Training a YOLO v2 Object Detector for the following object classes:

* Crack

Training on single CPU.
Initializing input data normalization.
```

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch RMSE	Mini-batch Loss	Base Learning Rate
1	1	00:00:15	5.03	25.3	0.0010
2	10	00:02:09	2.19	4.8	0.0010
3	20	00:04:12	1.64	2.7	0.0010
5	30	00:06:16	1.56	2.4	0.0010
6	40	00:08:21	1.19	1.4	0.0010
8	50	00:10:24	0.93	0.9	0.0010
9	60	00:12:21	1.05	1.1	0.0010
10	70	00:14:17	1.05	1.1	0.0010
12	80	00:16:15	1.14	1.3	0.0010
13	90	00:18:06	0.51	0.3	0.0010
15	100	00:19:56	0.84	0.7	0.0010
16	110	00:21:47	0.71	0.5	0.0010
18	120	00:23:36	0.68	0.5	0.0010
19	130	00:25:29	0.68	0.5	0.0010
20	140	00:27:19	0.55	0.3	0.0010

```
Detector training complete.
*************************************************************************
```

Figure 2.6: The training of object detection model YOLO v2

2.2.2.2 Unet and ResNet-18 Semantic Segmentation Model

As the semantic segmentation classifies each pixel in the image into its respective category, it takes more computational time during the training process. Thus, only 30 images are used for model training for both Unet and ResNet-18 semantic segmentation models. Each image is labelled by using pixel label function in the image labeller application. The crack region is labelled as one class while the normal wall is labelled as another class. Figure 2.7 illustrated the region labelling process. The crack region is labelled with red colour while the normal wall region is labelled with grey colour. This labelled dataset is saved as gTruthpixel matrix file so that it can be used for model training in the later part.

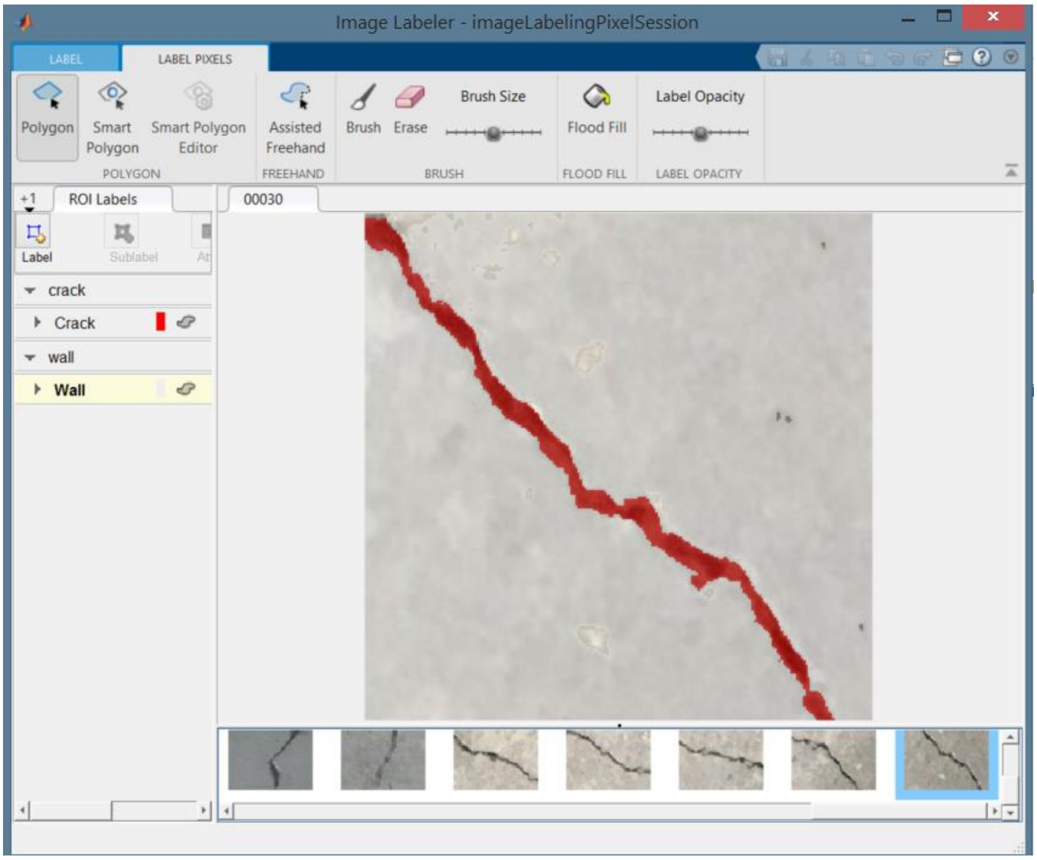


Figure 2.7: The segmentation region labeling process

After the labelling process, the labelled data can be used for both Unet and ResNet-18 model training.

```
data = load('gTruthpixel.mat');
imageDir = fullfile(data.gTruth.DataSource.Source);
labelDir = fullfile(data.gTruth.LabelData.PixelLabelData);

imds = imageDatastore(imageDir);
imds.ReadFcn = @customReadDatastoreImage;
classNames = [data.gTruth.LabelDefinitions.Name];
labelIDs    = [data.gTruth.LabelDefinitions.PixelLabelID];

pxds = pixelLabelDatastore(labelDir,classNames,labelIDs);
pxds.ReadFcn = @customReadDatastoreImage;
ds = pixelLabelImageDatastore(imds,pxds);
```

The code above loads both the labelled pixel and the crack images into matlab datastore. Then it is resized to 224×224 and combined as one variable named as ds. This ds will then be used for Unet and ResNet-18 model training.

```
15 -    imageSize = [224 224 3];
16 -    numClasses = 2;
17
18 -    lgraph = unetLayers(imageSize, numClasses);  % Unet model
19      % lgraph = deeplabv3plusLayers(imageSize, numClasses, "resnet18");% ResNet-18 model
20
21
22 -    options = trainingOptions('sgdm', ...
23          'InitialLearnRate',1e-3, ...
24          'MaxEpochs',20, ...
25          'VerboseFrequency',10);
26
27 -    net = trainNetwork(ds,lgraph,options);
```

The code above sets the criteria for model training. For Unet model training, line 18 is applied while line 19 is applied for ResNet-18 model training. The training method is set as sgdm, with a learning rate of 0.001, maximum epoch as 20 and the frequency of printing the progress of the training is set as 10. Line 27 starts the training process. The Unet trained model is saved as unetSegModel while the ResNet-18 model is saved as resnet18SegModel. Figure 2.8 and Figure 2.9 showed the training process of both models.

```
>> SemanticSegmentation
Training on single CPU.
Initializing input data normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:10:11 |       61.24% |       1.8188 |          0.0010 |
|      10 |          10 |       01:32:35 |       90.52% |       0.6297 |          0.0010 |
|      20 |          20 |       02:55:20 |       92.73% |       0.2767 |          0.0010 |
|========================================================================================|
```

Figure 2.8: The training of segmentation model unet

```
>> SemanticSegmentation
Training on single CPU.
Initializing input data normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:01:00 |       60.20% |       0.6790 |          0.0010 |
|      10 |          10 |       00:09:44 |       95.42% |       0.0828 |          0.0010 |
|      20 |          20 |       00:18:47 |       96.42% |       0.0638 |          0.0010 |
|========================================================================================|
```

Figure 2.9: The training of segmentation model ResNet-18

2.2.2.3 Localization and Extraction Models Evaluation

10 crack images with different shapes are used to test the performance of the 3 trained models.

```
yolo2net= load('yolov2Model2.mat');
unet= load('unetSegModel.mat');
rs18net= load('resnet18SegModel.mat');
```

The code above loads the 3 trained models into MATLAB so that it can be used for crack region extraction.

```
%YOLOv2
[bboxes,scores] = detect(yolo2net.detector,I);

% unet
C = semanticseg(I, unet.net);
C = C == 'Crack';
B = labeloverlay(I,C);

%rs18net
D = semanticseg(I, rs18net.net);
D = D == 'Crack';
E = labeloverlay(I,D);

if isempty(bboxes)
    F = imread('Test\Positive\fail.png');
else
    F = insertObjectAnnotation(I,'rectangle',bboxes,scores);
end
```

The code above deploys the 3 trained models to extract the crack region. For YOLO v2, if it is unable to detect the crack, an image showing failed to detect will be displayed. For both the Unet and ResNet-18 semantic segmentation models, if no crack is detected, the pixel in the picture will not be highlighted. The results of the evaluation are shown in Figure 2.9 to Figure 2.18. The performance of the three trained models is summarized in Table 2.1.

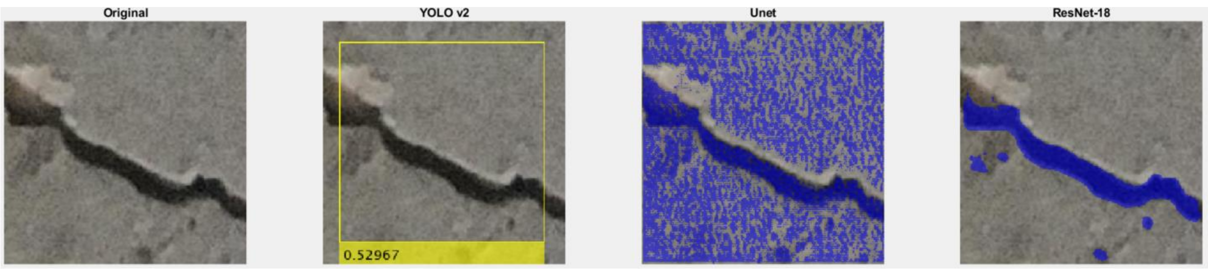


Figure 2.10: The performance of ROI extraction models in test set 01

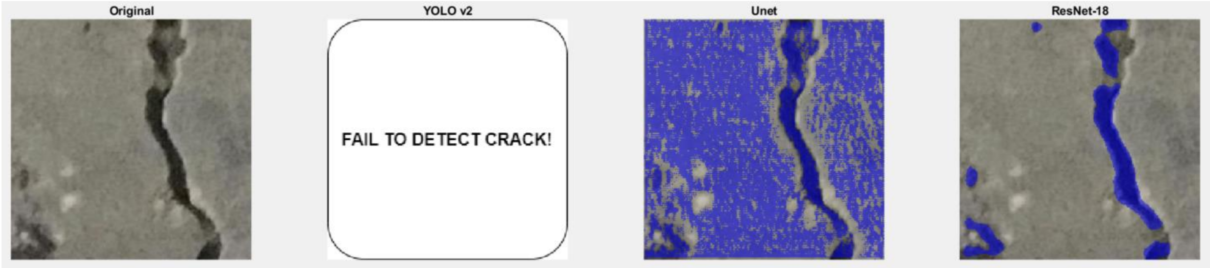


Figure 2.11: The performance of ROI extraction models in test set 02

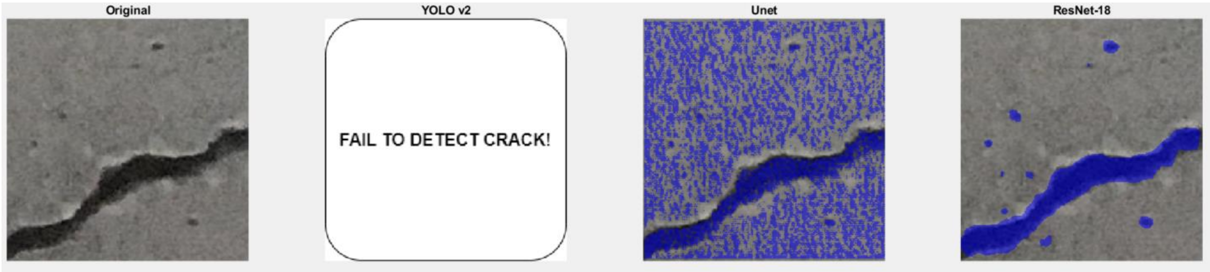


Figure 2.12: The performance of ROI extraction models in test set 03

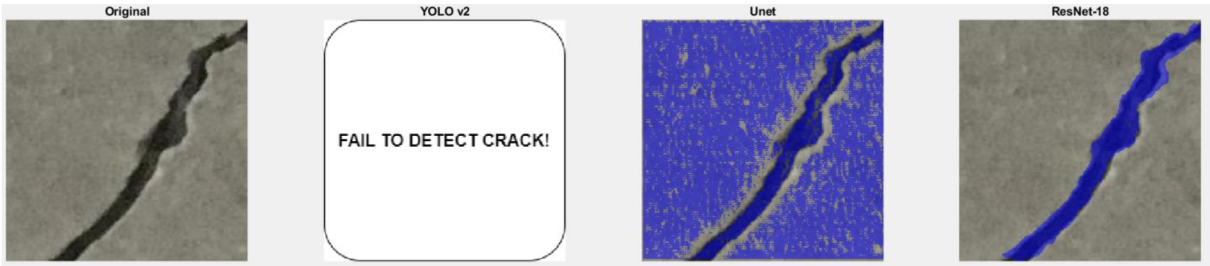


Figure 2.13: The performance of ROI extraction models in test set 04

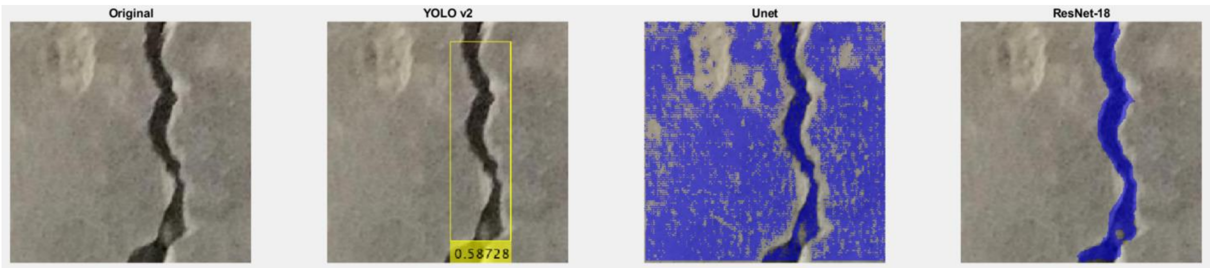


Figure 2.14: The performance of ROI extraction models in test set 05

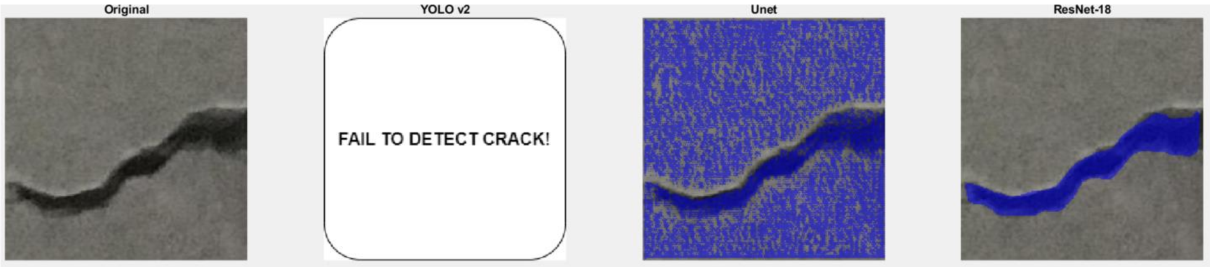


Figure 2.15: The performance of ROI extraction models in test set 06

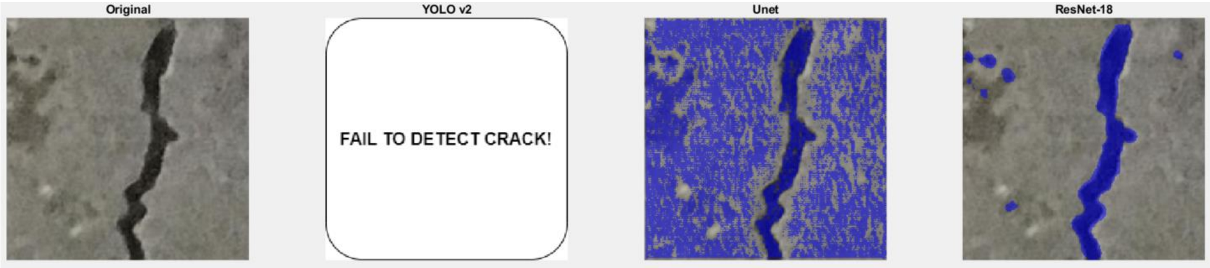


Figure 2.16: The performance of ROI extraction models in test set 07

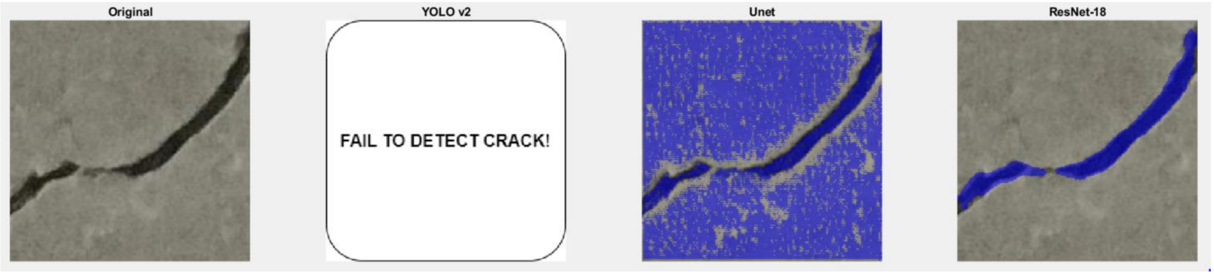


Figure 2.17: The performance of ROI extraction models in test set 08

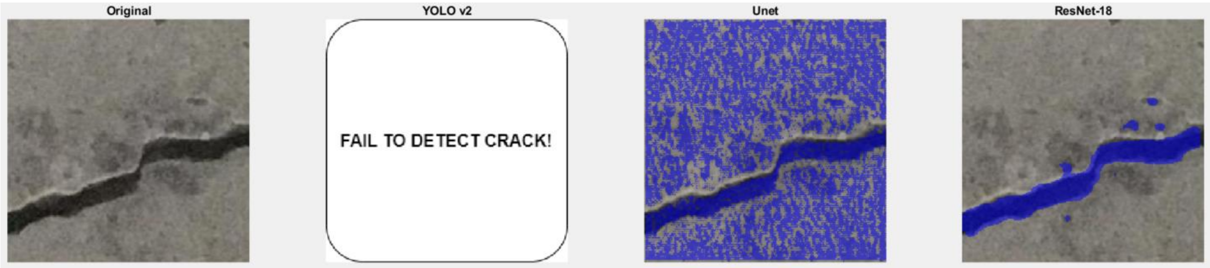


Figure 2.18: The performance of ROI extraction models in test set 09

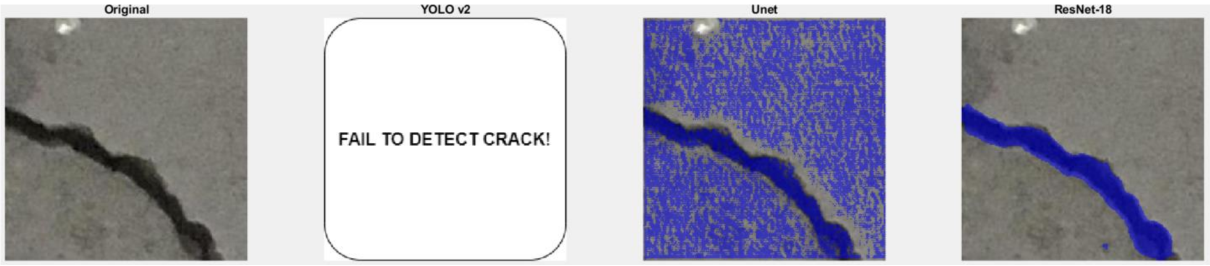


Figure 2.19: The performance of ROI extraction models in test set 10

Table 2.1: The performance of the 3 ROI Extraction Models

Performance	**YOLO v2 (Object Detector)**	**Unet (Semantic Segmentation)**	**ResNet-18 (Semantic Segmentation)**
Time taken for model training	27 mins 19 s	2 hours 55 mins 20 s	18 mins 47 s
Accuracy	$\frac{2}{10} \times 100\% = 20\%$	$\frac{0}{10} \times 100\% = 0\%$	$\frac{10}{10} \times 100\% = 100\%$

Even though Unet used the longest time for model training, it performed the worst as it is highlighting both the wall regions and the crack regions. YOLO v2 used intermediate time for model training and can only locate 2 out of 10 samples. Besides, for the 2 correctly extracted ROI samples, the confident level is only 0.53 and 0.59 respectively. This affirmed that object detector models which use aspect ratio to locate objects are not suitable to be applied for extracting objects which have varying shapes. Even though the ResNet-18 semantic segmentation model consumed the least time for model training, it performed the best among the 3 models. However, the model still falsely classifies some small regions as crack areas.

2.3 GUI Building Phase

The GUI of the system is developed by using the App Designer application in MATLAB. As the ResNet-18 segmentation model is showing the best performance, it is selected to localize and extract the crack area during the GUI integration. Figure 2.19 showed the process of designing the GUI by selecting and placing each component into the main frame.

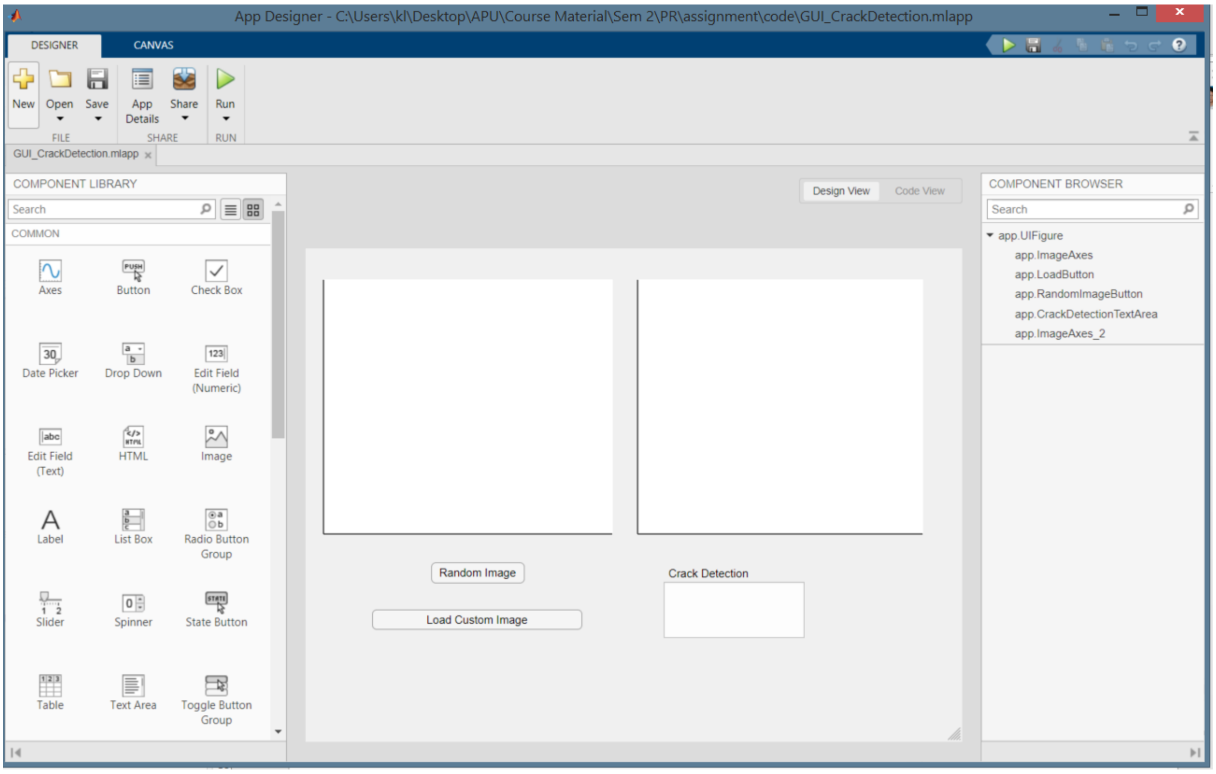


Figure 2.20: The process of GUI designing

```
        function startupFcn(app)
            % Configure image axes
            app.ImageAxes.Visible = 'off';
            app.ImageAxes_2.Visible = 'off';
            axis(app.ImageAxes, 'image');

            % Update the image and histograms
            updateimage(app, 'Test\Positive\00122.jpg');
        end
```

The code above sets the startup function to display one of the crack images from the folder.

```
        function LoadButtonPushed(app, event)

            % Display uigetfile dialog
            filterspec = {'*.jpg;*.tif;*.png;*.gif','All Image Files'};
            [f, p] = uigetfile(filterspec);

            % Make sure user didn't cancel uigetfile dialog
            if (ischar(p))
               fname = [p f];
               updateimage(app, fname);
            end
        end
```

The code above lets users select their desired image from the folder by clicking the load custom image button.

```
        function Random_img(app, event)
            fileList = dir(fullfile('Test\Positive', '/*.jpg'));
            randomIndex = randi(length(fileList), 1, 1); % Get random number.
            fullFileName = fullfile('Test\Positive', fileList(randomIndex).name);


            updateimage(app, fullFileName);
        end
    end
```

The code above will randomly select one image from the crack folder to be analysed when the user clicks the random image button.

```
    methods (Access = private)

        function updateimage(app,imagefile)


                try
                    im = imread(imagefile);
                catch ME
                    % If problem reading image, display error message
                    uialert(app.UIFigure, ME.message, 'Image Error');
                    return;
                end


                    % Display the original image
                    inputSize=[224 224];
                    im = imresize(im, inputSize);
                    imagesc(app.ImageAxes,im);

                    % Plot all histograms with the same data for grayscale
                    load model1.mat;
                    pred = classify (wallnet, im);
                    if string(pred) == "Positive"
                        load resnet18SegModel.mat;
                        C = semanticseg(im, net);
                        C = C == 'Crack';
                        B = labeloverlay(im,C);
                        imagesc(app.ImageAxes_2,B);
                        value = "Crack is detected. " + ...
                            " Crack Area is highlighted with Blue colour.";
                    else
                        value = "There is no Crack.";
                        imagesc(app.ImageAxes_2,im);
                    end
                    app.CrackDetectionTextArea.Value = value;


        end
    end
```

The code above will first classify the image into crack or non-crack category by using the trained wallnet model. If the image is classified as crack, the Resnet-18 segmentation model will be deployed to localize and extract the crack area. Then, the crack area will be displayed in the result frame and the crack status will be displayed on the crack message box. If there is no crack detected in the image, the original image will be displayed in the result frame and the message box will show “There is no Crack.”. The example of the GUI result when there is crack and when there is no crack is shown in Figure 2.4 and Figure 2.5 below.

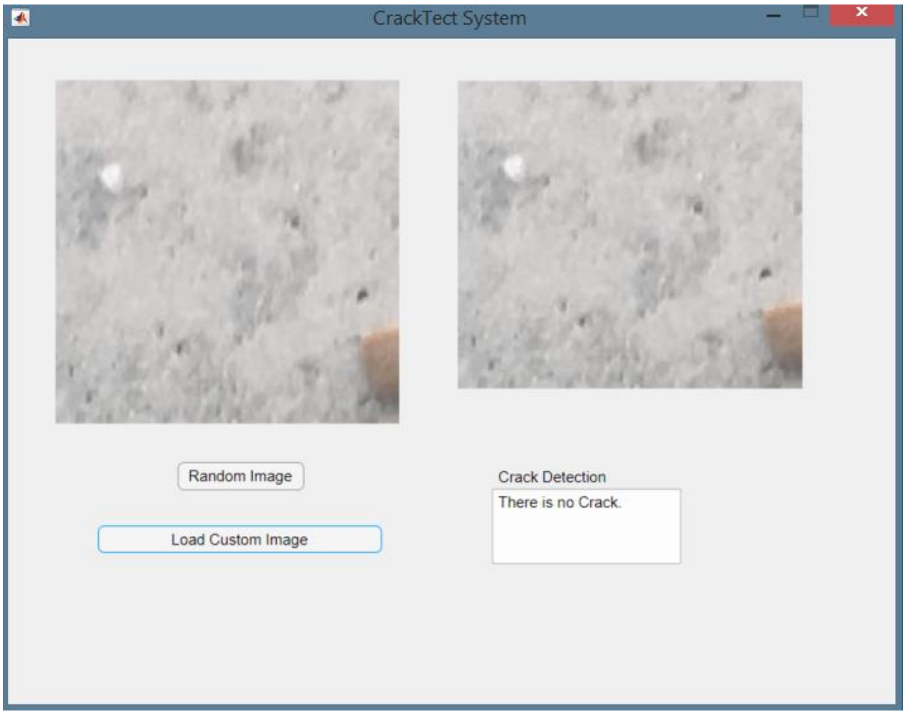


Figure 2.21: GUI when no crack is detected in the image.

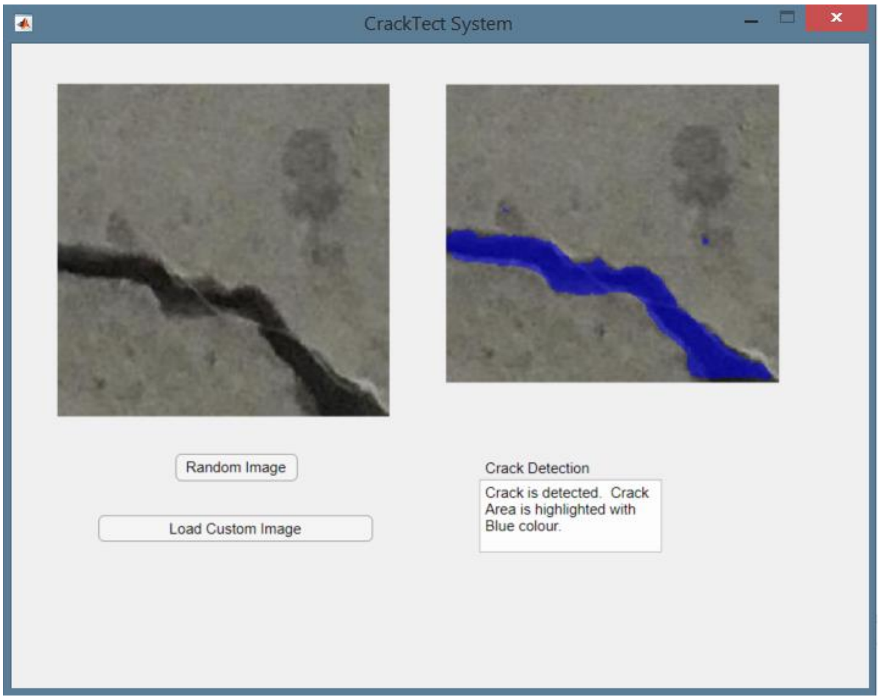


Figure 2.22: GUI when no crack is detected in the image.

SECTION 3

RESULTS

This section starts by presenting the system evaluation results, followed by discussing the strengths and limitations of the system.

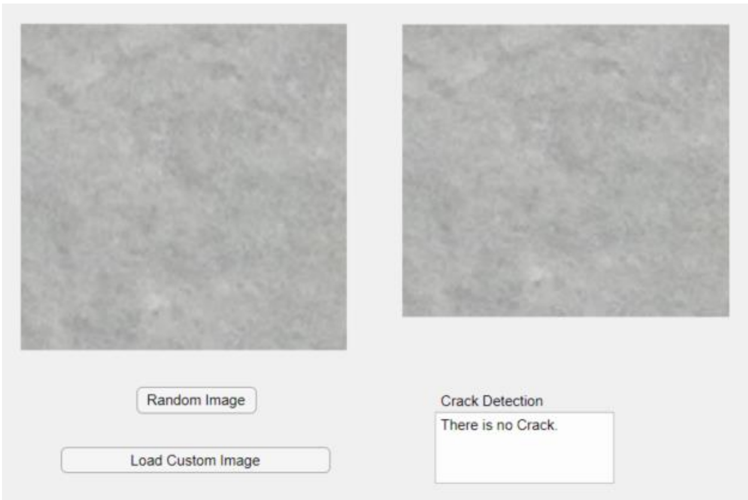
3.1 System Evaluation

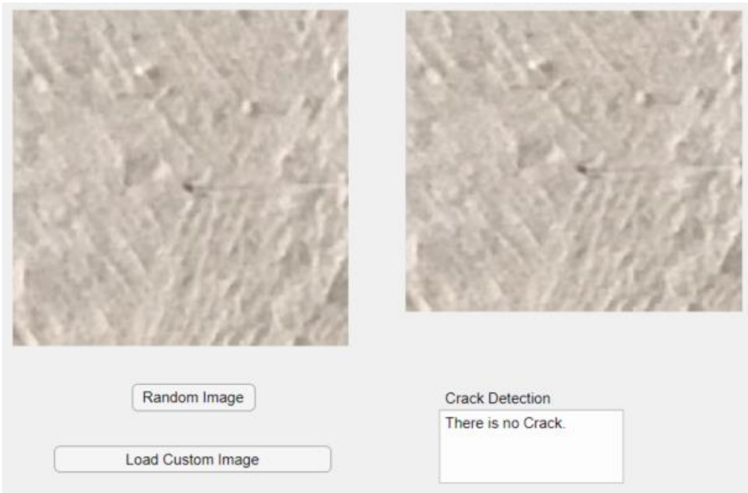
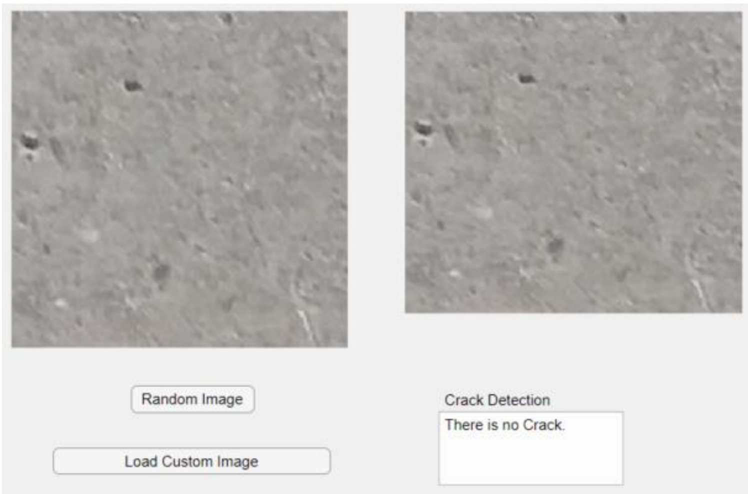
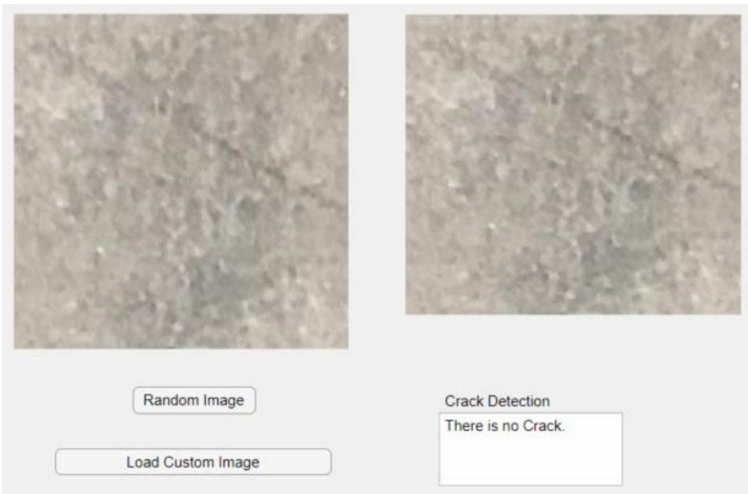
30 images from each class are used for evaluating the system. Only 10 samples from each class are selected and present in the form of screenshot results.

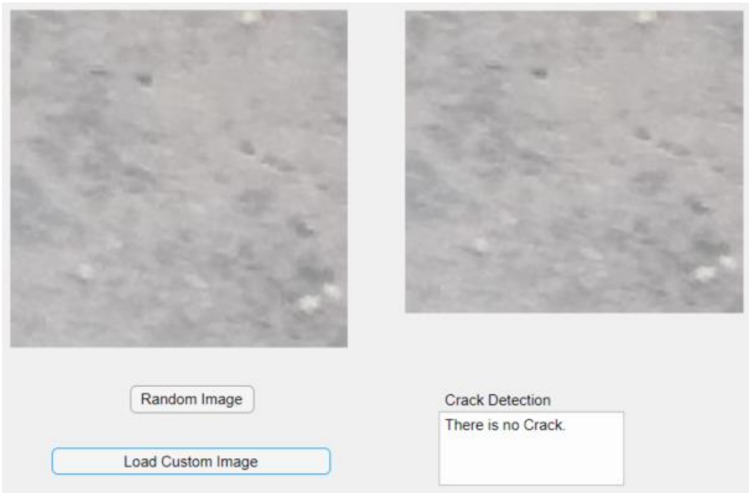
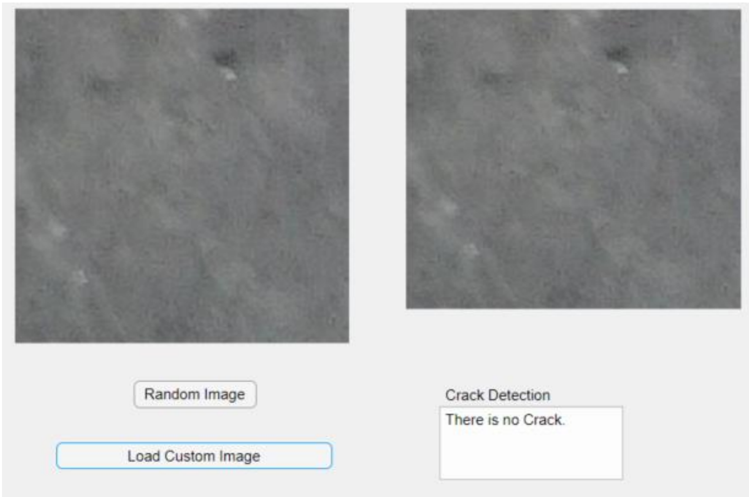
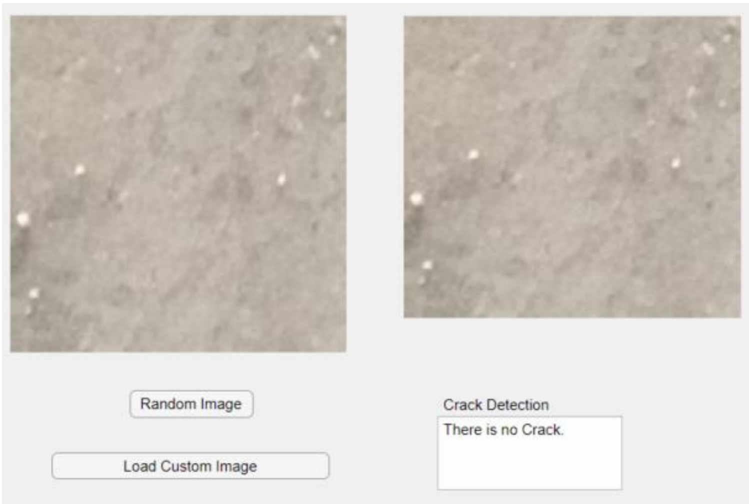
3.1.1 Examples of System Testing

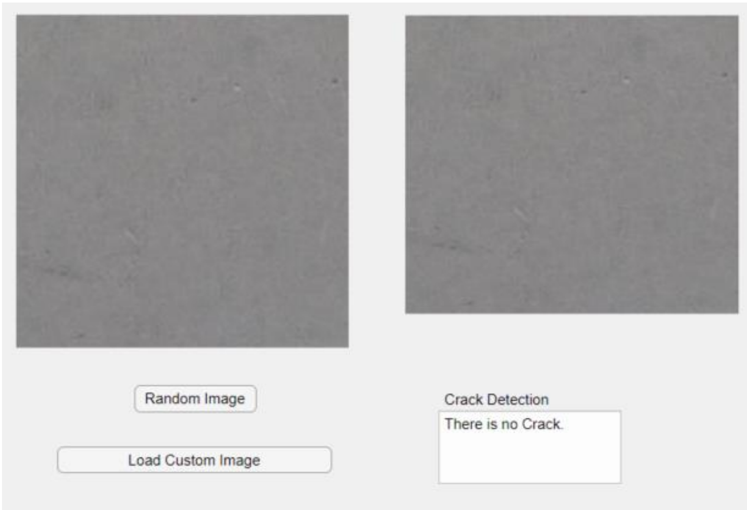
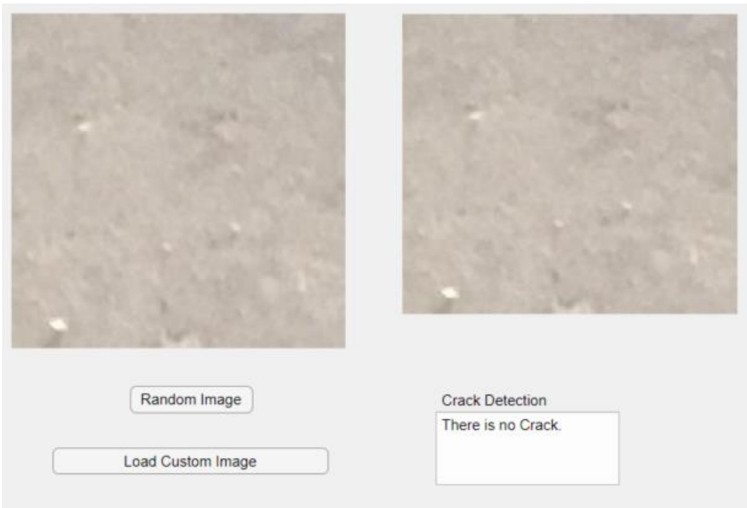
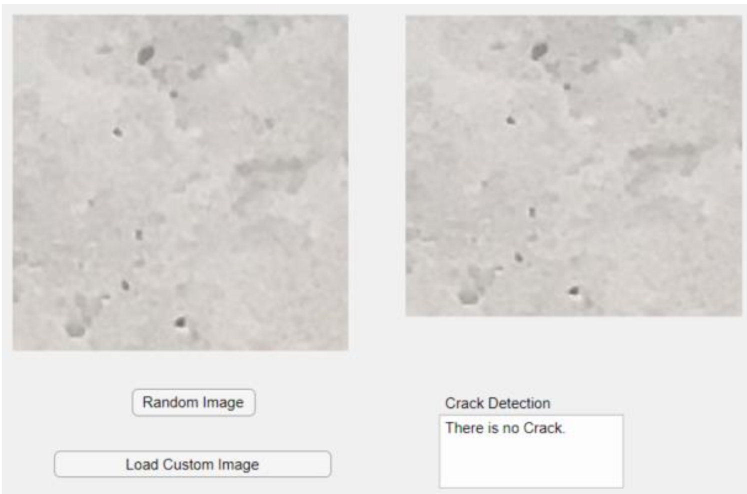
20 examples of system testing, 10 from each class are presented in the Table 3.1 below.

Table 3.1: Screenshots for the 20 examples of system testing

Examples	Screenshot Results
<u>EX 1:</u> True Class: Non-Crack Predicted Class: Non-Crack ROI Extraction: NA	

<table>
<tr>
<td><u>EX 2:</u>
True Class: Non-Crack
Predicted Class: Non-Crack
ROI Extraction: NA</td>
<td>
</td>
</tr>
<tr>
<td><u>EX 3:</u>
True Class: Non-Crack
Predicted Class: Non-Crack
ROI Extraction: NA</td>
<td>
</td>
</tr>
<tr>
<td><u>EX 4:</u>
True Class: Non-Crack
Predicted Class: Non-Crack
ROI Extraction: NA</td>
<td>
</td>
</tr>
</table>

<u>EX 5:</u> True Class: Non-Crack Predicted Class: Non-Crack ROI Extraction: NA	
<u>EX 6:</u> True Class: Non-Crack Predicted Class: Non-Crack ROI Extraction: NA	
<u>EX 7:</u> True Class: Non-Crack Predicted Class: Non-Crack ROI Extraction: NA	

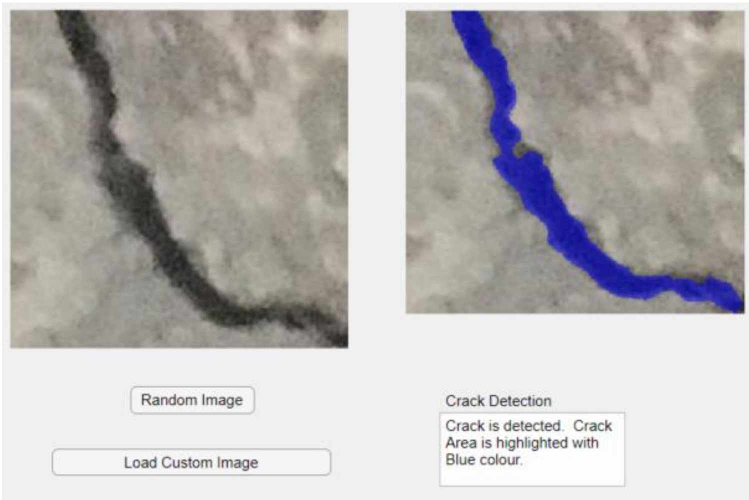
EX 8: True Class: Non-Crack Predicted Class: Non-Crack ROI Extraction: NA	
EX 9: True Class: Non-Crack Predicted Class: Non-Crack ROI Extraction: NA	
EX 10: True Class: Non-Crack Predicted Class: Non-Crack ROI Extraction: NA	

EX 11:

True Class: Crack

Predicted Class: Crack

ROI Extraction: Succeed

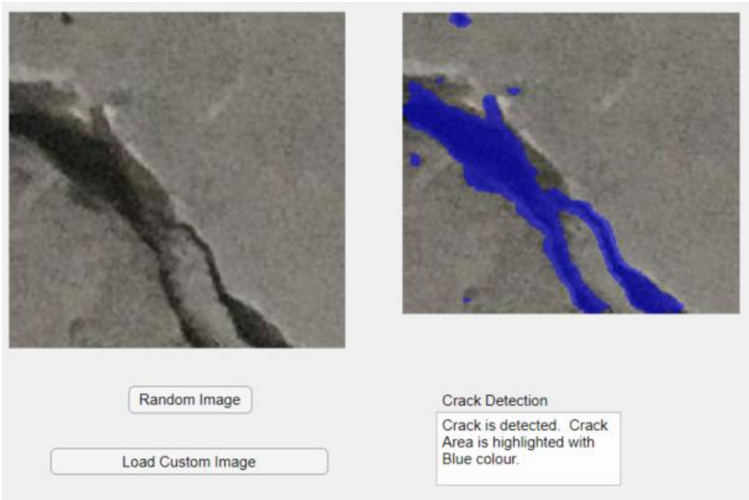


EX 12:

True Class: Crack

Predicted Class: Crack

ROI Extraction: Succeed

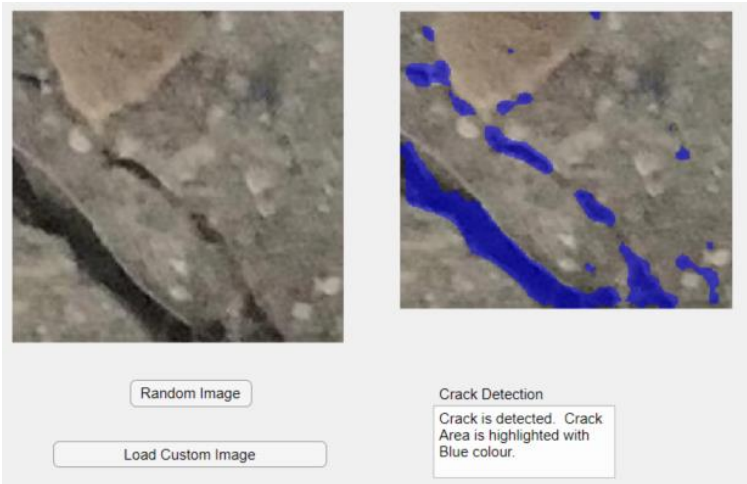
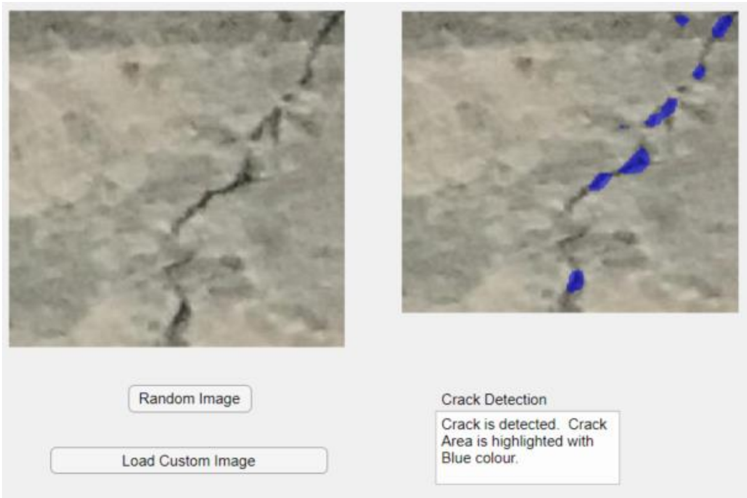
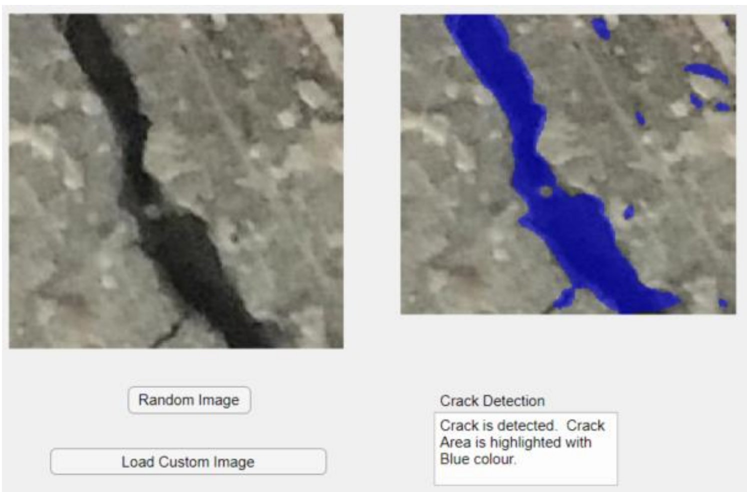


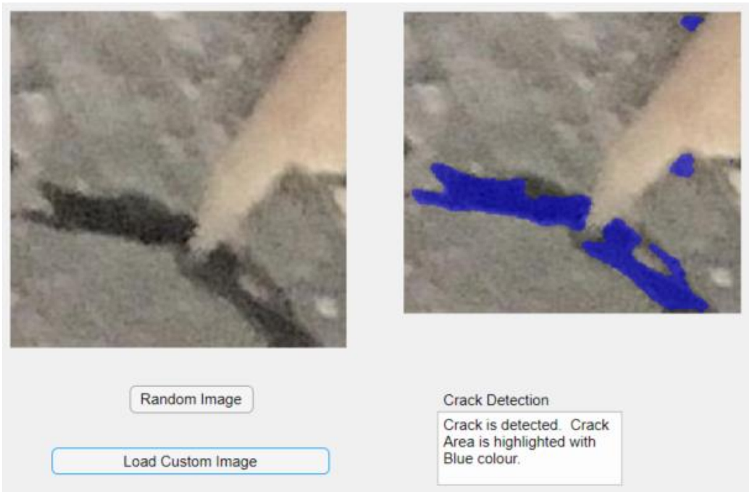
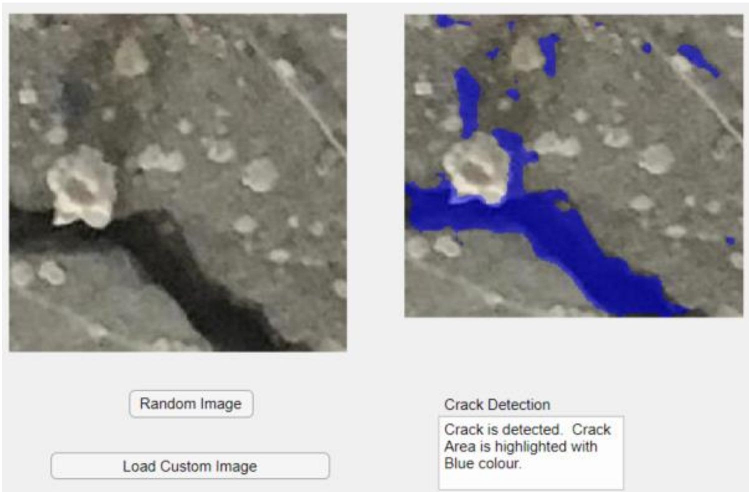
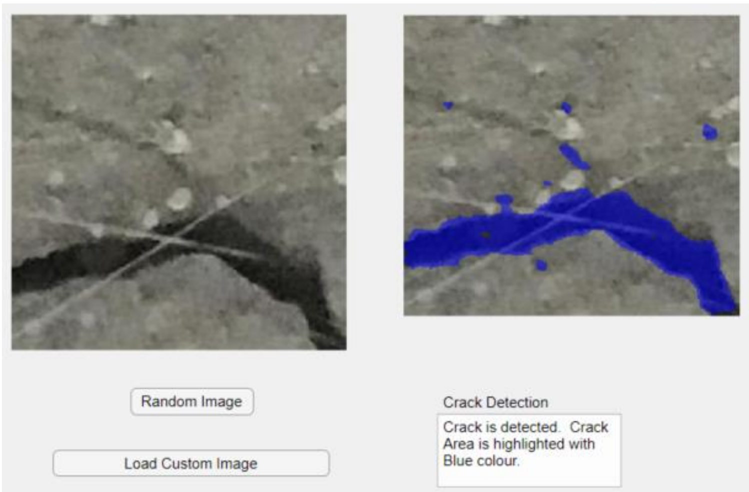
EX 13:

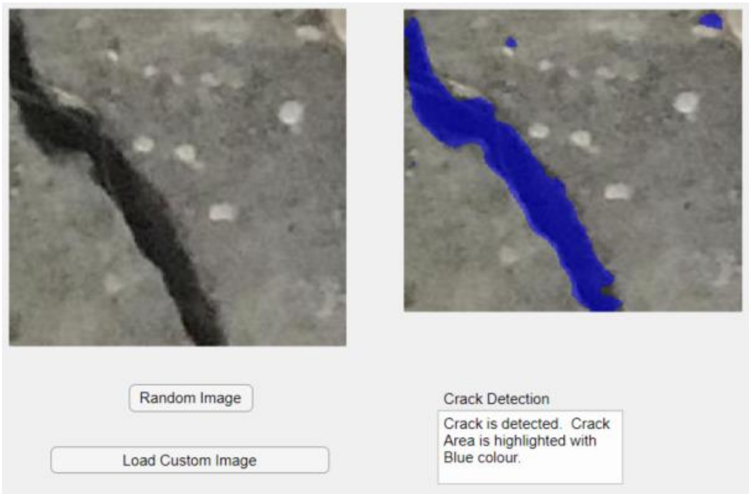
True Class: Crack

Predicted Class: Crack

ROI Extraction: Succeed

<u>EX 14:</u> True Class: Crack Predicted Class: Crack ROI Extraction: Succeed	
<u>EX 15:</u> True Class: Crack Predicted Class: Crack ROI Extraction: Succeed	
<u>EX 16:</u> True Class: Crack Predicted Class: Crack ROI Extraction: Succeed	

<u>EX 17:</u> True Class: Crack Predicted Class: Crack ROI Extraction: Succeed	
<u>EX 18:</u> True Class: Crack Predicted Class: Crack ROI Extraction: Succeed	
<u>EX 19:</u> True Class: Crack Predicted Class: Crack ROI Extraction: Succeed	

<u>EX 20:</u> True Class: Crack Predicted Class: Crack ROI Extraction: Succeed	

3.1.2 Performance of Classification and Region of Interest (ROI) Extraction

60 images are used for evaluating the classification performance of the system, including the 20 examples shown in the previous section. For the performance of ROI extraction, only the 30 crack images are involved. This is because the localization and extraction model will only be deployed in the second stage if the classification of the image is Crack class. The summary of the system performance is shown in Table 3.2.

Table 3.2: Performance of the system in classification and ROI extraction

Classification			**ROI Extraction**	
TP	30		Total Error	30
FP	0		Correct	30
TN	30		Accuracy	1
FN	0			
Accuracy	1			
Precision	1			
Recall	1			
F1 Score	1			

3.2 Strengths and Limitations

The system is very good at identifying non-crack images and crack images as can be seen in the accuracy, precision, recall and F1 score value, all with 100% for the 60 test images. This indicated that the first stage classification model wallnet is well trained in classifying non-crack and crack images. Besides, the system is also good at extracting the crack region. For all the images that are classified as crack, the ROI extraction model using ResNet-18 is able to localize and extract the crack regions. Moreover, the system can also detect and extract crack regions from images that are not from the collected dataset and identify irrelevant images as non-crack images as shown in Figure 3.1 and Figure 3.2 below.

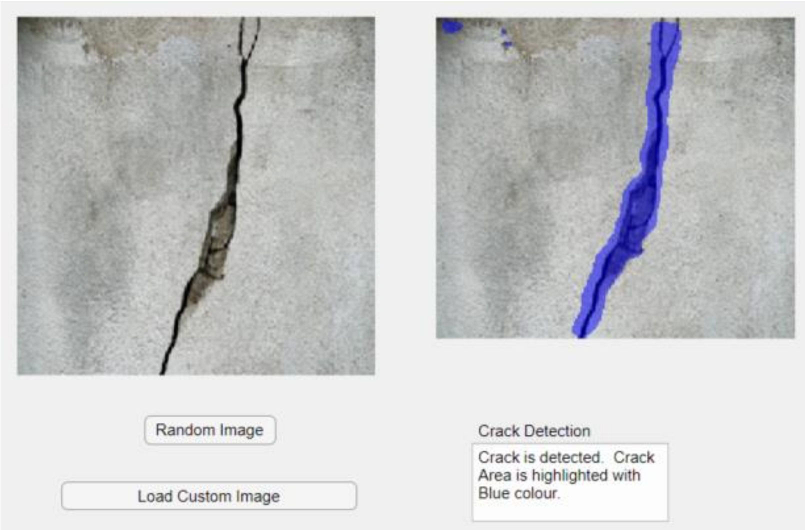


Figure 3.1: System testing on random image obtained from google image seach

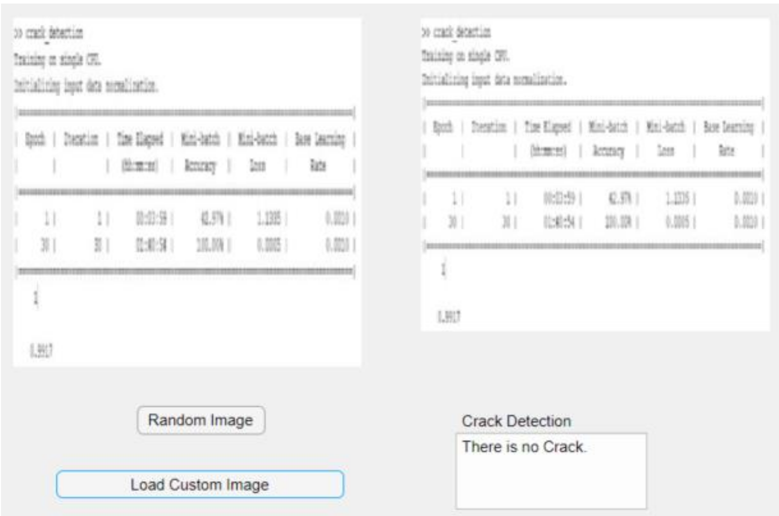


Figure 3.2: System testing on random image that does not contain crack

The weakness of the model is that the system will still classify some of the crack images as non-crack images, resulting in false negatives while 1 non-crack image is classified as crack image, resulting in false positive as shown in Figure 2.2. This indicated that the first stage classification model wallnet is not perfect in identifying crack images, especially those images that have a lot of noises, where some other objects or dirt are present in the images. Other than that, the ROI extraction model is not able to extract all the crack regions in the crack image, especially those regions that are not salient enough or when the crack is too small. It also falsely classified some small regions as crack areas, especially those dark regions. There is still room for improvement for both the classification model and the ROI extraction model.

SECTION 4

CONCLUSIONS

The 2 stages CrackTect System developed in this study is able to perform well in both of the image classification and the ROI extraction processes. The system is divided into 2 stages, the crack classification stage for crack detection and the ROI extraction stage for the localization of crack area. This is done to reduce the processing time for analysing non-crack images and also to reduce the frequency of falsely ROI extraction. Longer training time and more complex network does not always produce a better result as can be seen in the case where the ResNet-18 is performing better than Unet even though it uses way less time for model training. For localizing and extracting objects with irregular shapes, like the crack region in this study, semantic segmentation type of model is a better choice compared to object detector type of model as the aspect ratio of these objects are not fixed. Aspect ratio is a crucial feature used in object detector based models.

REFERENCES

Çağlar Fırat Özgenel (2019) *Mendeley Data - Concrete Crack Images for Classification*. Available at: https://data.mendeley.com/datasets/5y9wdsg2zt/2 (Accessed: 13 December 2020).

Liu, Z. *et al.* (2019) 'Computer vision-based concrete crack detection using U-net fully convolutional networks', *Automation in Construction*. Elsevier, 104(April), pp. 129–139. doi: 10.1016/j.autcon.2019.04.005.

Lopes, U. K. and Valiati, J. F. (2017) 'Pre-trained convolutional neural networks as feature extractors for tuberculosis detection', *Computers in Biology and Medicine*. Elsevier Ltd, 89, pp. 135–143. doi: 10.1016/j.compbiomed.2017.08.001.

Mandal, V., Uong, L. and Adu-Gyamfi, Y. (2019) 'Automated Road Crack Detection Using Deep Convolutional Neural Networks', in *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*. Institute of Electrical and Electronics Engineers Inc., pp. 5212–5215. doi: 10.1109/BigData.2018.8622327.

Redmon, J. and Farhadi, A. (2017) 'YOLO9000: Better, faster, stronger', in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Institute of Electrical and Electronics Engineers Inc., pp. 6517–6525. doi: 10.1109/CVPR.2017.690.

Szegedy, C. *et al.* (2015) 'Going deeper with convolutions', in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 1–9. doi: 10.1109/CVPR.2015.7298594.

Zhang, J. *et al.* (2019) 'Concrete Cracks Detection Based on FCN with Dilated Convolution', *Applied Sciences*. MDPI AG, 9(13), p. 2686. doi: 10.3390/app9132686.

APPENDICE

Code for building classification wallnet:

```
wallds = imageDatastore("Crack_detection_samples","IncludeSubfolders",...
    true,"LabelSource","foldernames");


[trainImgs,testImgs] = splitEachLabel(wallds,0.6);

numClasses = numel(categories(wallds.Labels));

net = googlenet;
lgraph = layerGraph(net);

newFc = fullyConnectedLayer(2,"Name","new_fc");
lgraph = replaceLayer(lgraph,"loss3-classifier",newFc);
newOut = classificationLayer("Name","new_out");
lgraph = replaceLayer(lgraph,"output",newOut);


options = trainingOptions("sgdm","InitialLearnRate", 0.001);

testLabels = testImgs.Labels;
inputSize=[224 224];
trainImgs = augmentedImageDatastore(inputSize, trainImgs);
testImgs = augmentedImageDatastore(inputSize, testImgs);

[wallnet,info] = trainNetwork(trainImgs, lgraph, options);

testpreds = classify(wallnet,testImgs);



disp(nnz(testpreds == testLabels)/ numel(testpreds));

confusionchart(testLabels,testpreds);

%% Check result
testds = imageDatastore("Test","IncludeSubfolders",...
    true,"LabelSource","foldernames");
tLabels = testds.Labels;
tImgs = augmentedImageDatastore(inputSize, testds);
tpreds = classify(wallnet,tImgs);

disp(nnz(tpreds == tLabels)/ numel(tpreds));

confusionchart(tLabels,tpreds);
```

Code for choosing optimal anchor box:

```
data = load('Gtruth.mat');
crackDataset = data.gTruth.LabelData;
data.gTruth.DataSource.Source;
crackDataset(1:4,:)

allBoxes = vertcat(crackDataset.Crack{:});
aspectRatio = allBoxes(:,3) ./ allBoxes(:,4);
area = prod(allBoxes(:,3:4),2);

figure
scatter(area,aspectRatio)
```

```
xlabel("Box Area")
ylabel("Aspect Ratio (width/height)");
title("Box Area vs. Aspect Ratio")

trainingData = boxLabelDatastore(crackDataset(:,:));

numAnchors = 5;
[anchorBoxes,meanIoU] = estimateAnchorBoxes(trainingData,numAnchors);

maxNumAnchors = 15;
meanIoU = zeros([maxNumAnchors,1]);
anchorBoxes = cell(maxNumAnchors, 1);
for k = 1:maxNumAnchors
    % Estimate anchors and mean IoU.
    [anchorBoxes{k},meanIoU(k)] = estimateAnchorBoxes(trainingData,k);
end

figure
plot(1:maxNumAnchors,meanIoU,'-o')
ylabel("Mean IoU")
xlabel("Number of Anchors")
title("Number of Anchors vs. Mean IoU")
```

<u>Code for building YOLO v2 model:</u>

```
data = load('Gtruth.mat');
lblBox = boxLabelDatastore(data.gTruth.LabelData);
imPath = imageDatastore(data.gTruth.DataSource.Source);
crackData = combine(imPath, lblBox);
scaledData = transform(crackData,@scaleGT);

anchorBoxes = estimateAnchorBoxes(scaledData,10);

%% showing one label image example
data = read(crackData);
I = data{1};
bbox = data{2};
annotatedImage = insertShape(I,'Rectangle',bbox);
annotatedImage = imresize(annotatedImage,2);
figure
imshow(annotatedImage)

%%
net = resnet18;
numClasses = 1;
imageSize = [224 224 3];

lgraph = yolov2Layers(imageSize,numClasses,anchorBoxes,net,...
    "res5b_relu","ReorgLayerSource","res3a_relu");

options = trainingOptions('sgdm', ...
        'MiniBatchSize',16, ....
        'InitialLearnRate',1e-3, ...
        'MaxEpochs',20,...
        'VerboseFrequency',10);

detector = trainYOLOv2ObjectDetector(scaledData,lgraph,options);

I = imread('Crack_detection_samples\Positive\00100.jpg');
I = imresize(I,imageSize(1:2));
```

```
[bboxes,scores] = detect(detector,I);

I = insertObjectAnnotation(I,'rectangle',bboxes,scores);
figure
imshow(I)



function data = scaleGT(data)
    targetSize = [224 224];
    % data{1} is the image
    scale = targetSize./size(data{1},[1 2]);
    data{1} = imresize(data{1},targetSize);
    % data{2} is the bounding box
    data{2} = bboxresize(data{2},scale);
end
```

<u>Code for building Unet and ResNet-18 semantic segmentation model:</u>

```
data = load('gTruthpixel.mat');
imageDir = fullfile(data.gTruth.DataSource.Source);
labelDir = fullfile(data.gTruth.LabelData.PixelLabelData);

imds = imageDatastore(imageDir);
imds.ReadFcn = @customReadDatastoreImage;
classNames = [data.gTruth.LabelDefinitions.Name];
labelIDs   = [data.gTruth.LabelDefinitions.PixelLabelID];

pxds = pixelLabelDatastore(labelDir,classNames,labelIDs);
pxds.ReadFcn = @customReadDatastoreImage;
ds = pixelLabelImageDatastore(imds,pxds);


imageSize = [224 224 3];
numClasses = 2;

lgraph = unetLayers(imageSize, numClasses);  % Unet model
% lgraph = deeplabv3plusLayers(imageSize, numClasses, "resnet18");% ResNet-
18 model


options = trainingOptions('sgdm', ...
    'InitialLearnRate',1e-3, ...
    'MaxEpochs',20, ...
    'VerboseFrequency',10);

net = trainNetwork(ds,lgraph,options);


I = imread('Test\Positive\00121.jpg');
I = imresize(I,imageSize(1:2));
imshow(I)
C = semanticseg(I, net);
C = C == 'Crack';
B = labeloverlay(I,C);
figure
montage({I,B});
```

```
function data = customReadDatastoreImage(filename)
% code from default function:
onState = warning('off', 'backtrace');
c = onCleanup(@() warning(onState));
data = imread(filename); % added lines:
data = imresize(data,[224 224]);
end
```

Code for ROI Extraction models evaluation:

```
yolo2net= load('yolov2Model2.mat');
unet= load('unetSegModel.mat');
rs18net= load('resnet18SegModel.mat');
% analyzeNetwork(net);

imageSize = [224 224 3];

%Read different images

% I = imread('Crack_detection_samples\Positive\00100.jpg');
I = imread('Test\Positive\00262.jpg');
% I = imread('Test\Positive\00259.jpg');
% I = imread('Test\Positive\00206.jpg');
% I = imread('Test\Positive\00252.jpg');
% I = imread('Test\Positive\00278.jpg');
% I = imread('Test\Positive\00263.jpg');
% I = imread('Test\Positive\00251.jpg');
% I = imread('Test\Positive\00240.jpg');
% I = imread('Test\Positive\00236.jpg');
% I = imread('Test\Positive\00128.jpg');

I = imresize(I,imageSize(1:2));


%YOLOv2
[bboxes,scores] = detect(yolo2net.detector,I);

% unet
C = semanticseg(I, unet.net);
C = C == 'Crack';
B = labeloverlay(I,C);

%rs18net
D = semanticseg(I, rs18net.net);
D = D == 'Crack';
E = labeloverlay(I,D);

if  isempty(bboxes)
    F = imread('Test\Positive\fail.png');
else
    F = insertObjectAnnotation(I,'rectangle',bboxes,scores);
end

figure
subplot(141)
imshow(I), title('Original');
```

```
subplot(142)
imshow(F), title('YOLO v2');
subplot(143)
imshow(B), title('Unet');
subplot(144)
imshow(E), title('ResNet-18');
```