PROJECT TITLE:

DETECTING BUILDING DEFECTS USING VVG16 AND IBM WATSON

INTRODUCTION

Overview:

Building defects is one of the major components of building problems that significantly needed attention.

When a building fails to function as it should, we must immediately seek for detection. For this we have built CNN(Convolution Neural Network) model, where it is provided with dataset.

Here data preprocessing and model building applications such as importing necessary libraries, Pre-trained CNN model ,Pre-trained CNN model ,Training and testing the model .Save the Model are done.

Then we installed flask and requests .We then included three html pages named home, intro, upload where home is the home page intro page is the introduction upload page is used to display the output. We have also created app.py which is python scripting file.

Purpose:

The main aim or idea behind this project is to detect the defects before it causes the damage.

These defects can significantly affect the structural integrity and the aesthetic aspect of buildings to prevent this we are using CNN pre-trained model VGG16 to analyze the type of

building defect on the given parameters.

The objective of the project is to build an application to detect the type of building defect.

LITERATURE SURVEY

Existing Problem and methods to solve this:

As the time passes many buildings due to poor maintainance develops defects like cracks, flakes etc., if left undetected it can cause hazardous damages.

Traditional methods for this type of work commonly comprise of engaging building surveyors to undertake a condition assessment which involves a lengthy site inspection to produce a systematic recording of the physical condition of the building elements, including cost estimates of immediate and projected long-term costs of renewal, repair and maintenance of the building.

Proposed Solution:

In this project detecting building defects such as cracks , flakes and roof defects, We are using CNN pre-trained model VGG16 to analyze the type of building defect on the given parameters.

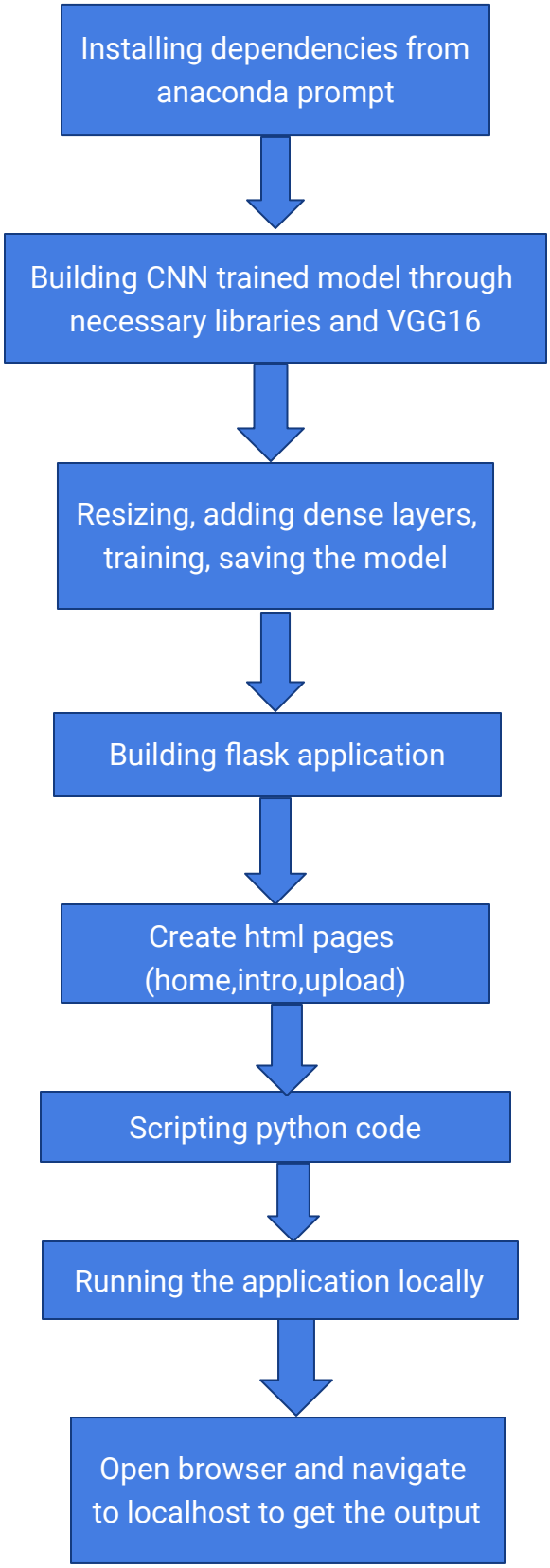
The objective of the project is to build an application to detect the type of building defect.

The model uses an integrated webcam to capture the video frame and the video frame is compared with the Pre-trained model

The type of building defect is identified and showcased on the OpenCV window and emergency pull is initiated.

THEORITICAL ANALYSIS:

Block Diagram:



Software requirements:

In our project we used softwares like :

*Anaconda prompt - Anaconda is a distribution of Python.Its is free and open-source and makes package management and deployment simpler.it has tools to easily collect data from sources using ML and AI,it has good community support and it is the industry standard for developing,testing and training on a single machine

*jupyter notebook -Jupyter is a free, open-source, interactive web tool known as a computational notebook, which researchers can use to combine software code, computational output, explanatory text and multimedia resources in a single document. *Notepad- We used this software to write and execute html codes.

*Spyder - Spyder,the Scientific Python Development Environment,is a free integrated development environment(IDE) that is included with Anaconda.It includes editing,interactive testing,debugging and introspection features.

* Chrome - Used to execute localhost. * IBM Watson- We trained the model on IBM .

Hardware requirements:

The hardware requirements for our project are :
* System Type - 64-bit Operating System
*RAM-8GB
*processor-11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GH

EXPERIMENTAL ANALYSIS

While working on the solution we investigated what actually flask is and as our project needed three html web pages, we saw few videos on how to create web pages using html and cnn on youtube.

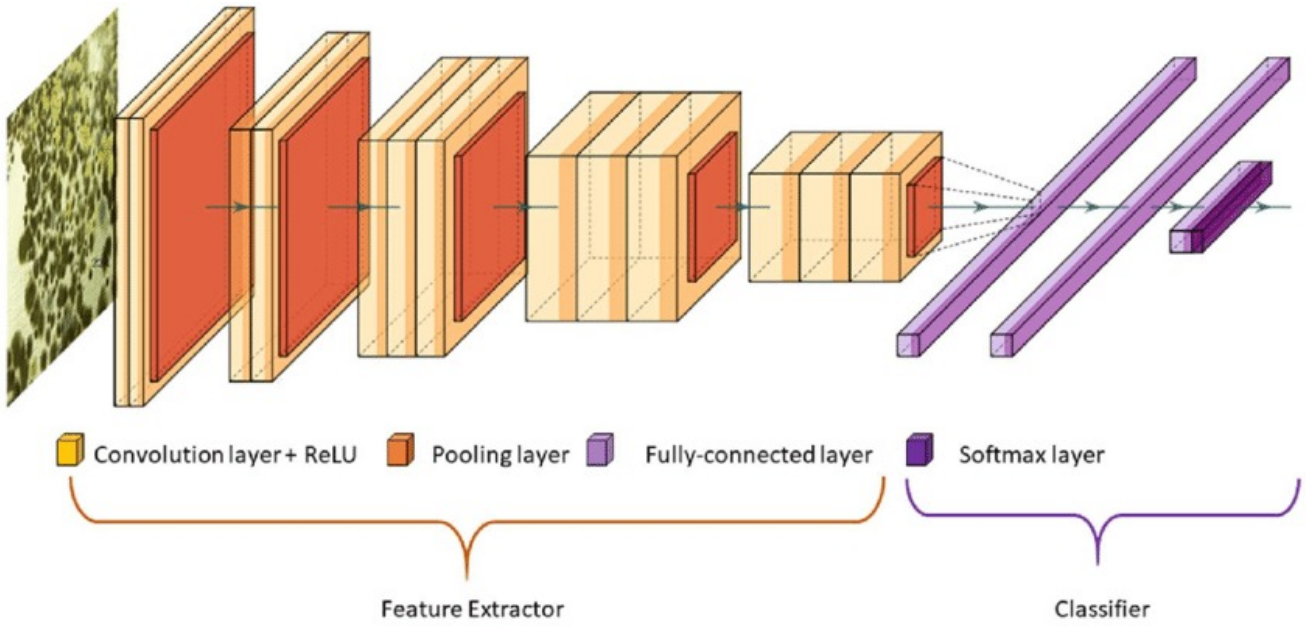
Before this we first noted down the aim of project and blue print of the project in order to gain knowledge about them through internet, books etc .

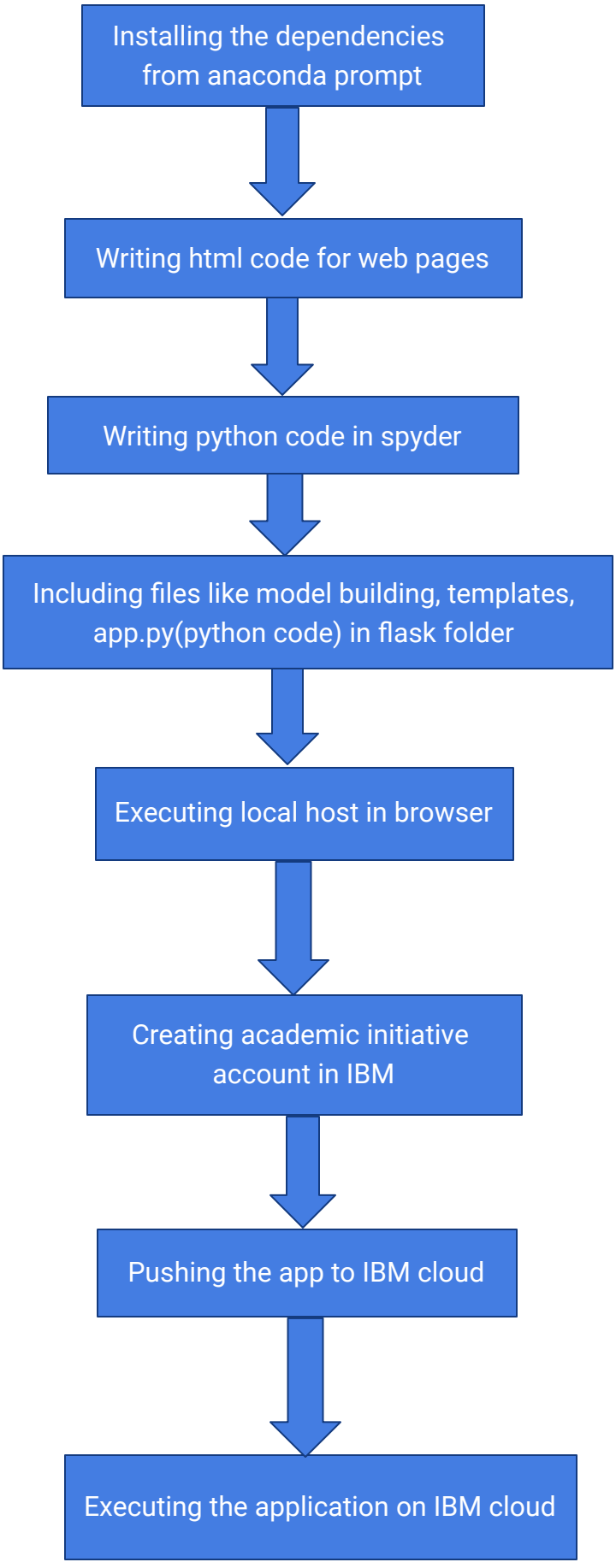
In html, we learnt about adding buttons, adding images, adding text, adding navbar and etc so as to include them in our pages. We then learnt how to write code for flask python scripting file and testing file with the VGG16 by seeing some videos and also by referring some books.

As our project is a flask application, in order to gain knowledge about flask we watched videos that are provided in our guided project itself .And at last in order to push the app onto IBM, we saw video that our guided project provided us.

FLOW CHART

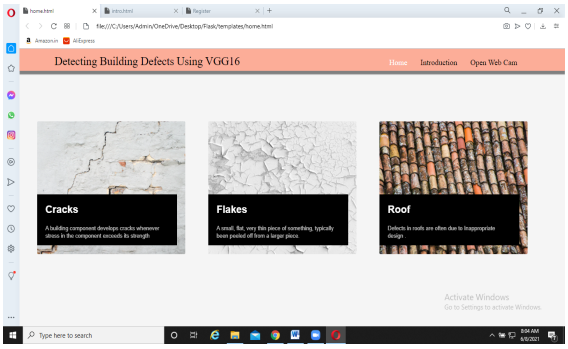
Drawing showing the control flow of the solution:




Installing the dependencies from anaconda prompt
Writing html code for web pages
Writing python code in spyder
Including files like model building, templates, app.py(python code) in flask folder
Executing local host in browser
Creating academic initiative account in IBM
Pushing the app to IBM cloud
Executing the application on IBM cloud

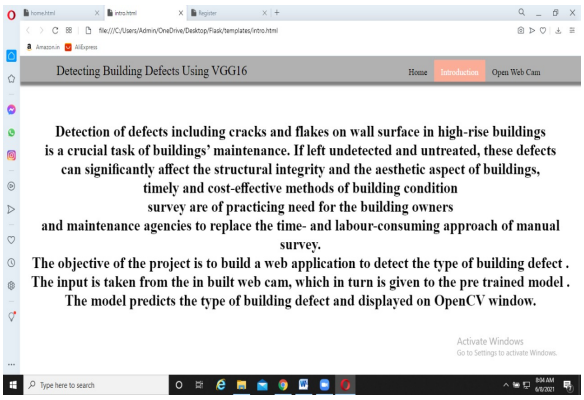
RESULT

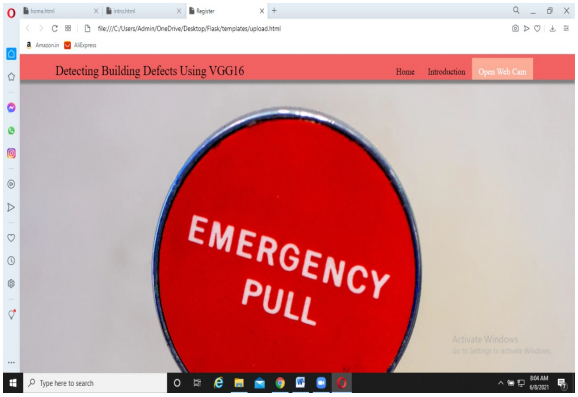
* We finally created three web pages in order to translate text from one language to the other language.
* In the output language section we included executing local host in browser.
* An image is given as an input from the web cam and the output is in the form of text whether it is cracks, flakes or roof.
* Here are the screenshot images of our project.



home.html

intro.html

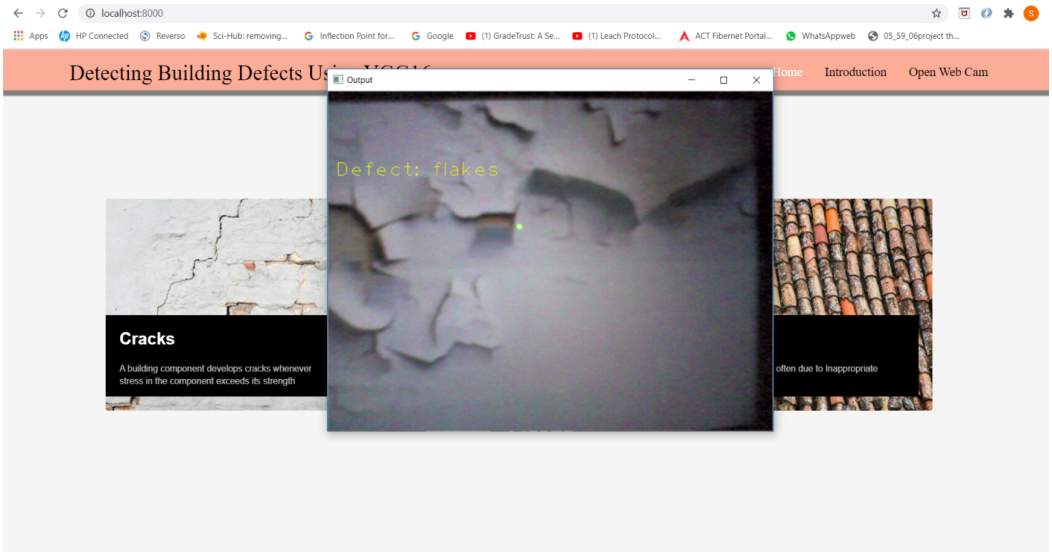


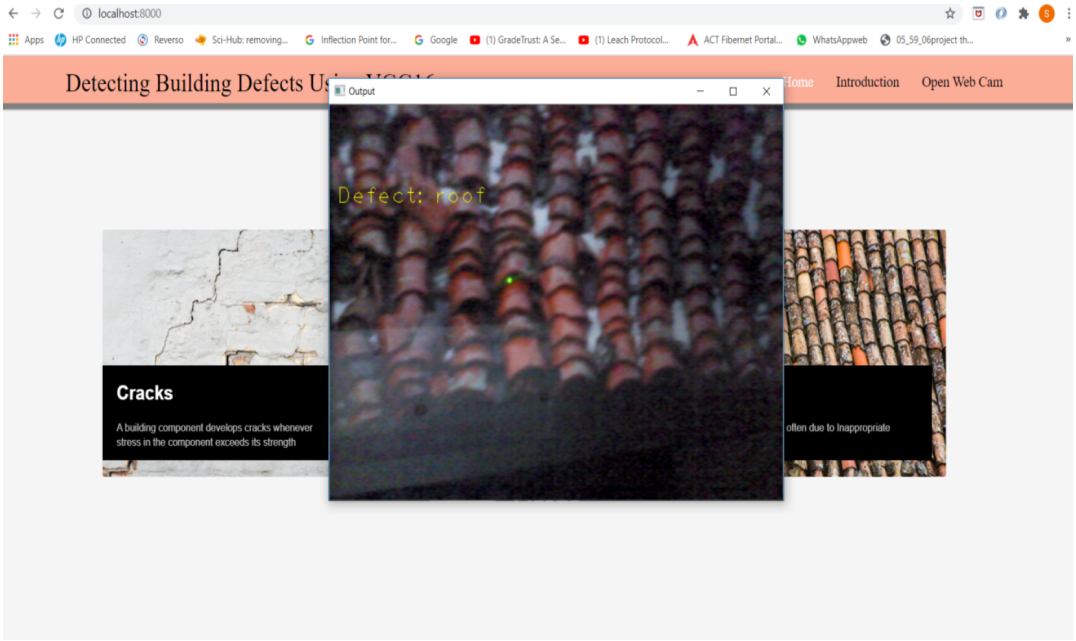


upload.html

These are the screenshots of the output of our project:

* When we have given the wall images through our webcam as well as our disc.
* The output is displayed in text form.





ADVANTAGES

* Defect tracking helps you ensure that bugs found in the system actually get fixed. Sure, it's great for testers and developers to have a conversation and to recreate the problem together. If the problem gets fixed immediately, great! Maybe it doesn't need to be logged. But if it just gets put down somewhere on a mental to-do list, there's a good chance it'll slip through the cracks.

* Defect tracking tools not only provide a way to ensure follow-through but also provide valuable metrics. Depending on the tool being used, the team can tie defects to changed code, tests, or other data that will allow for traceability or analysis on defect trends. If a certain module is riddled with defects, it may be time to review and rewrite that module.

* Defect tracking tools allow for a repository of documentation that will provide value for troubleshooters or for support personnel later on if there's a workaround for an issue. Having a tool in place also sends notifications to the right people when a bug needs to be fixed, tested, or marked as resolved.

DISADVANTAGES

* Many of the disadvantages of defect tracking have more to do with the overhead of the processes and tools than the idea of defect tracking itself. Some organizations use multiple tools to track defects of different types, and those tools often don't integrate well with one another.

* Complications can arise out of confusion over descriptions, lack of information, tools that are overly cumbersome and require mandatory fields for which the user doesn't have the answers, and difficulty in reporting. Sometimes the process overhead required to open a bug is more time-consuming than simply fixing the bug.

* If every low-priority defect is tracked, the defect report will include many bugs that will never be fixed because fixing them doesn't provide a high enough ROI. However, keeping them in a defect report as "open" will require constant reanalysis and may imply poor quality, when in fact these defects aren't issues customers care about.

APPLICATIONS

* Thermography methodologies for detecting energy related building defects

* Time-lapse thermography for building defect detection

* Improving the detection of thermal bridges in buildings via on-site infrared thermography

* Geophysical methods for defect detection in architectural structures.

CONCLUSION

Defect detection using computer vision models started to pick up popularity in the 21st century, as the object detection models became more and more popular. The exponential growth in popularity of deep learning methods for defect detection and other computer vision related applications in recent years is fueled by lots of researchers plunging into this sector, as well as by hardware and data breakthroughs. Lots of areas of defect detection solutions were reviewed in this paper and as demonstrated deep learning methods achieve state-of-the-art performance in defect detection, while also having great generalization properties.

The general accepted idea is that the dataset, as well as the chosen model lead to great performances, thus both need to receive attention from the developer. The training hardware doesn‘t need to be expensive, if the application does not mandate it. Smaller, more specific applications can yield great results and thus, improve the workings of a small lab or business just by using general purpose laptops and generic detection models which will be tweaked for the defects we look into. While general applications that have a target as to detect lots of defects, need very large and balanced datasets, a hardware setup with lots of computational power and a specific detection model that is not just tweaked for defect detection, but built from the ground up for the specific action that we want it to perform.

Thus, hardware setup and availability(that‘s why recommended is using virtual machines on a desktop machine learning station for general solutions) plays a role in the performance of the models. All in all, there are multiple factors that need to be taken into consideration when

planning to design defect detection algorithms.

There is no rule of thumb when choosing which object detection model shall generalize the best on the particular dataset of interest. First of all, the developer needs to be in constant contact with the system technician or quality inspector, in order to assure that the dataset is of great quality and is reliable.

FUTURE SCOPE

For the future works, the challenges and limitation that we were facing this in paper will be addressed. The presented paper had to set a number of limitations, i.e., firstly, multiple types of the defects are not considered at once.

This means that the images considered by the model belonged to only one category. Secondly, only the images with visible defects are considered.

Thirdly, consideration of the extreme lighting and orientation, e.g., low lighting, too bright images, are not included in this study. In the future works, these limitations will be considered to be able to get closer to the concept of a fully automated detection.

Through fully satisfying these challenges and limitations, our present work will be evolved into a software application to perform real-time detection of defects using vision sensors including drones. The work will also be extended to cover other models that can detect other defects in construction such as cracks, structural movements, spalling and corrosion.

Our long-term vision includes plans to create a large, open source database of different building and construction defects which will support world-wide research on condition assessment of built assets.

BIBILOGRAPHY

1. Mohseni, H.; Setunge, S.; Zhang, G. M.; Wakefield, R. In Condition monitoring and condition aggregation for optimised decision making in management of buildings, Applied Mechanics and

Materials, 2013; Trans Tech Publ: 2013; pp 1719-1725.

2. Agdas, D.; Rice, J. A.; Martinez, J. R.; Lasa, I. R., Comparison of visual inspection and structuralhealth monitoring as bridge condition assessment methods. Journal of Performance of Constructed Facilities 2015, 30, (3), 04015049.

3. Shamshirband, S.; Mosavi, A.; Rabczuk, T., Particle swarm optimization model to predict scour depth around bridge pier. arXiv preprint arXiv:1906.08863 2019.

4. Zhang, Y.; Anderson, N.; Bland, S.; Nutt, S.; Jursich, G.; Joshi, S., All printed strain sensors: Building blocks of the aircraft structural health monitoring system. Sensors and Actuators A: Physical 2017, 253, 165-172.

5. Noel, A. B.; Abdaoui, A.; Elfouly, T.; Ahmed, M. H.; Badawy, A.; Shehata, M. S., Structural health monitoring using wireless sensor networks: A comprehensive survey. IEEE Communications Surveys & Tutorials 2017, 19, (3), 1403-1423.

APPENDIX

Source code:

This is the code which we have developed.

```
# import the necessary packages
from flask import Flask,render_template
# Flask-It is our framework which we are going to use to run/serve our application.
#request-for accessing file which was uploaded by the user on our application.
import cv2 # opencv library
from tensorflow.keras.models import load_model#to load our trained model
import numpy as np
#import os from tensorflow.keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image
'''
def playaudio(text):
    speech=gTTS(text)
    print(type(speech))
    speech.save("output1.mp3")
```

```
  playsound("output1.mp3")
  return
'''
app = Flask(__name__,template_folder="templates")
# initializing a flask app
# Loading the model model=load_model('model_building_defects_vgg16.h5')
print("Loaded model from disk")
#app=Flask(__name__,template_folder="templates")
@app.route('/', methods=['GET'])
def index():
    return render_template('home.html')
@app.route('/home', methods=['GET'])
def home():
    return render_template('home.html')
@app.route('/intro', methods=['GET'])
def about():
return render_template('intro.html')
@app.route('/upload', methods=['GET', 'POST'])
def predict():
   # Get a reference to webcam #0 (the default one)

      print("[INFO] starting video stream...")
      vs = cv2.VideoCapture(0)
      #writer = None
      (W, H) = (None, None)
      #loop over frames from the video file stream
      while True:

        # read the next frame from the file
        (grabbed, frame) = vs.read()
        # if the frame was not grabbed, then we have reached the end
        # of the stream
          if not grabbed:
              break
        # if the frame dimensions are empty, grab them
          if W is None or H is None:
             (H, W) = frame.shape[:2]
# clone the output frame, then convert it from BGR to RGB
        # ordering and resize the frame to a fixed 64x64
        output = frame.copy()
        #print("apple")
```

```
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            frame = cv2.resize(frame, (224, 224))
            #frame=image.img_to_array(frame)
            #frame = frame.astype("float32")
            x=image.img_to_array(frame)
            x=np.expand_dims(frame, axis=0)
            img_data=preprocess_input(x)
    # clone the output frame, then convert it from BGR to RGB
           # ordering and resize the frame to a fixed 64x64
              output = frame.copy()
              #print("apple")
              frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
              frame = cv2.resize(frame, (224, 224))
              #frame=image.img_to_array(frame)
              #frame = frame.astype("float32")
              x=image.img_to_array(frame)
              x=np.expand_dims(frame, axis=0)
              img_data=preprocess_input(x)
              result = np.argmax(model.predict(img_data), axis=-1)
              index=['crack','flakes','roof']
              result=str(index[result[0]])
              #print(result)
              #result=result.tolist()
              cv2.putText(output, "Defect: {}".format(result), (10, 120), cv2.FONT_HERSHEY_PLAIN, 2,
    (0,255,255), 1)
              #playaudio("Emergency it is a disaster")
              cv2.imshow("Output", output)
              key = cv2.waitKey(1) & 0xFF
              # if the `q` key was pressed, break from the loop
              if key == ord("q"):
                  break
     # release the file pointers
     print("[INFO] cleaning up...")
     vs.release()
     cv2.destroyAllWindows()
     return render_template("upload.html")
     if __name__ == '__main__':
            app.run( port=8000, debug=False)
```