Detecting Building Defects Using VGG16 With IBM Cloud

INDEX:

b. UI output

1. INTRODUCTION :

Detection of defects including cracks and flakes on the wall surfaces in high-rise buildings is a crucial task of buildings maintenance. If left undetected and untreated, these defects can signi?cantly a?ect the structural integrity and the aesthetic aspect of buildings, Time and cost-e?ective methods of building condition survey are of practicing need for the building owners and maintenance agencies to replace the time- and labor-consuming approach of the manual survey.

Clients are increasingly looking for fast and effective means to quickly and frequently survey and communicate the condition of their buildings so that essential repairs and maintenance work can be done in a proactive and timely manner before it becomes too dangerous and expensive. 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.

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

Overview:

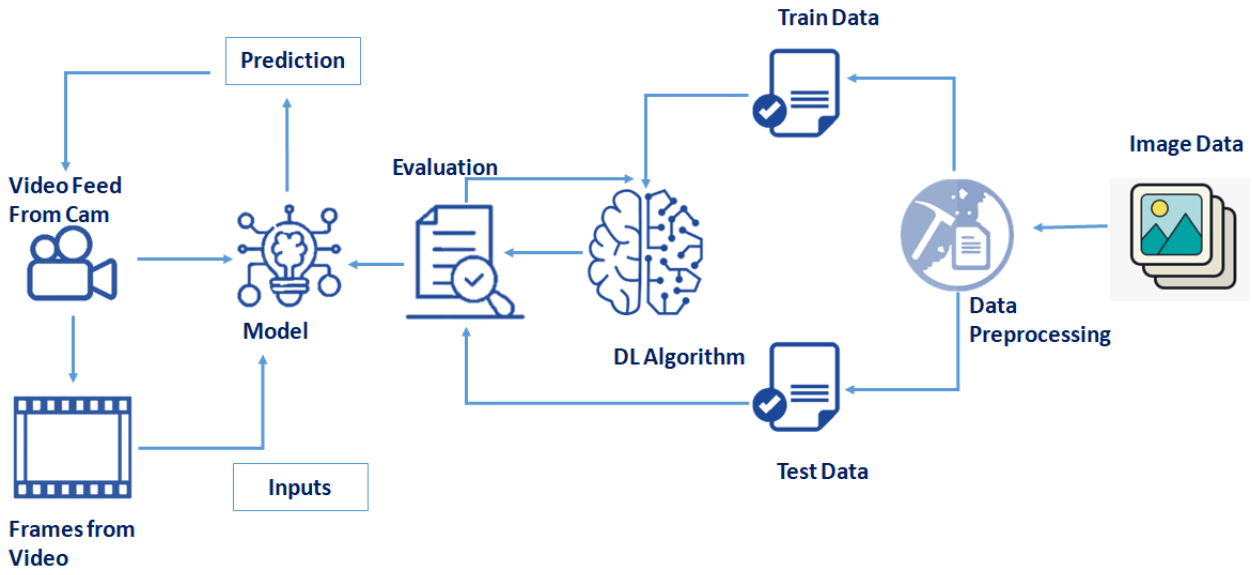
1. Defifining our classififiaction categories

2. Collect training images

3. Train the model

4. Test our mode

2. Technical Architecture :



3. PROJECT OBJECTIVE :

By the end of this project you will:

- know fundamental concepts and techniques of the Artificial Neural Network and Convolution Neural Networks

- Gain a broad understanding of image data.

- Work with Sequential type of modelling

- Work with Keras capabilities
- Work with image processing techniques
- know how to build a web application using the Flask framework.

3. PROJECT FLOW :

- The user interacts with the UI (User Interface) to open the integrated webcam.
- The video frames are captured and analyzed by the model which is integrated with the flask application.
- Once the model analyses the video frames, the prediction is showcased on the UI and OpenCV window.

- **Data Collection**.
    - Collect the dataset or Create the dataset
- **Data Preprocessing**.
    - Import the ImageDataGenerator library
    - Configure ImageDataGenerator class
    - ApplyImageDataGenerator functionality to Trainset and Testset
- **Model Building**
    - Import the model building Libraries
    - Resizing the images
    - Pre-trained CNN model as a Feature Extractor
    - Adding Output Layer
    - Configure the Learning Process

- Training and testing the model
- Save the Model
- Test The model

- **Application Building**
  - Create an HTML file
  - Build Python Code

4. LITERATURE SURVEY :

- TO complete this project, you must require following software's, concept and packages

● **Anconda navigator :**

Refer to the link below to download anaconda navigator

● **Python packages :**

- open anaconda prompt as administrator
- Type “pip install numpy” and click enter.
- Type “pip install pandas” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type “pip install opencv-contrib-python” and click enter.
- Type “pip install tensorflow==2.3.0” and click enter.
- Type “pip install keras==2.4.0” and click enter.
- Type “pip install Flask” and click enter.

5. DATASET COLLECTION :

Artificial Intelligence is a data hunger technology, it depends heavily on data, without data, it is impossible for a machine to learn. It is the most crucial aspect that makes algorithm training possible. In Convolutional Neural Networks, as it deals with images, we need training and testing data set. It is the actual data set used to train the model for performing various actions. In this activity let's focus on gathering the dataset.

6. IMAGE PREPROCESSING :

Image Pre-processing includes the following main tasks

- **Import ImageDataGenerator Library.**

- **Configure ImageDataGenerator Class.**
- **Applying ImageDataGenerator functionality to the trainset and test set**

7. MODEL BUILDING :

- IMPORTING THE MODEL BUILDING LIBRARIES
- RESIZING THE IMAGES
- PRETRAINED CNN MODEL HAS A FEATURE EXTRACTOR
- ADDING DENSE LAYER
- CONFIGURE LEARNING PROCESS
- TRAIN THE MODEL
- SAVE THE MODEL TEST THE MODEL
- APPLICATION BUILDING
  - Create an HTML file
  - Build Python Code

8. ADVANTAGES & DISADVANTAGES :

Advantages:

● Very high accuracy in image recognization problems.● Automatically detects the important features without any human supervisions.

● Weight sharing

Disadvantages:

● CNN do not encode the position and orientation of object.

● Lack of ability to be spatially invariant to the input data

● Lots of training data is required

9.-APPLTCATIONS :

- Decoding Facia Recognization

- Analyzing Documents
- Historic and Environmental Collections
- Understanding Climate
- Grey Areas
- Advertising
- Other Interesting Fields

10. BIBILOGRAPHY :

We used saw some Reference videos in You Tube.

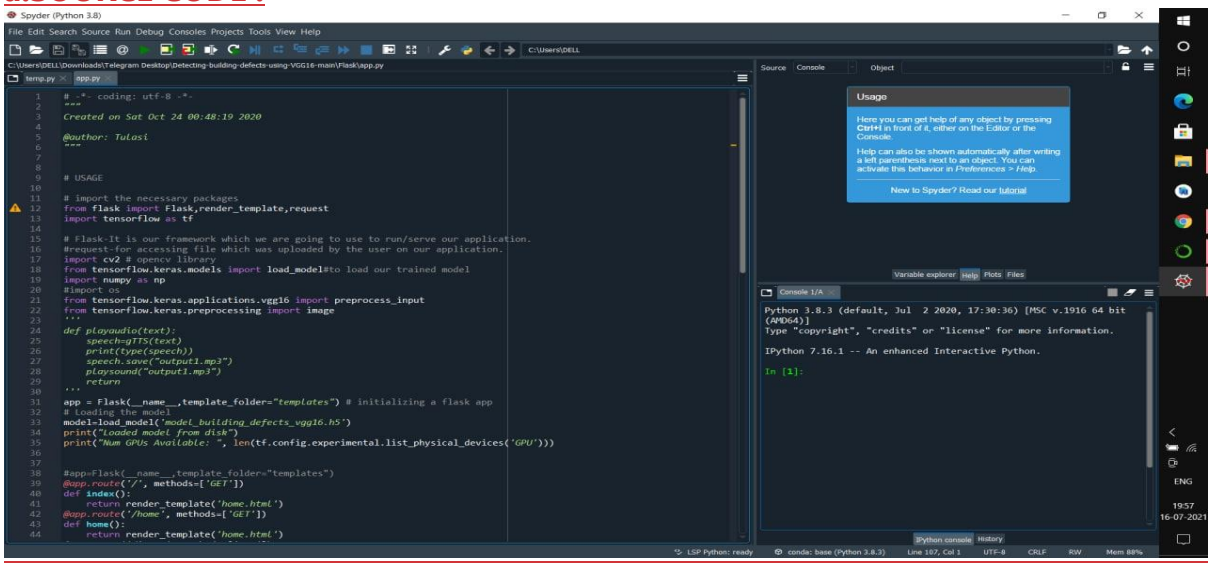
https://www.youtube.com/watch?v=DKSZHN7jftI

https://www.youtube.com/watch?v=aJ9wUDBoLUs

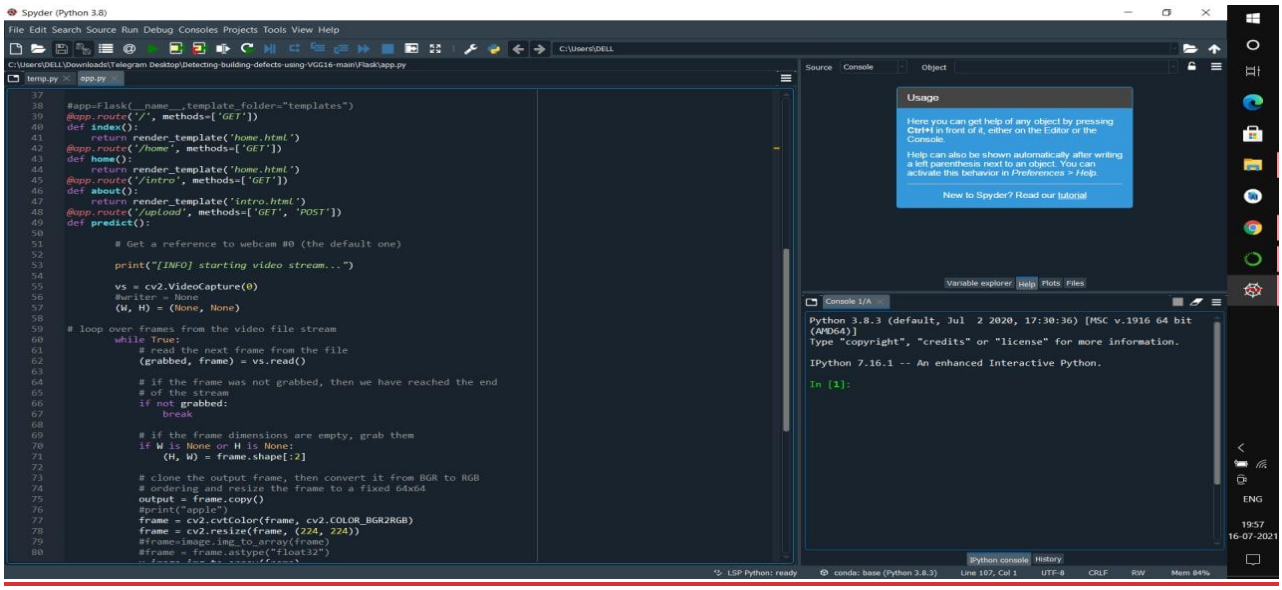
https://www.youtube.com/watch?v=lj4I_CvBnt0

11. APPENDIX :

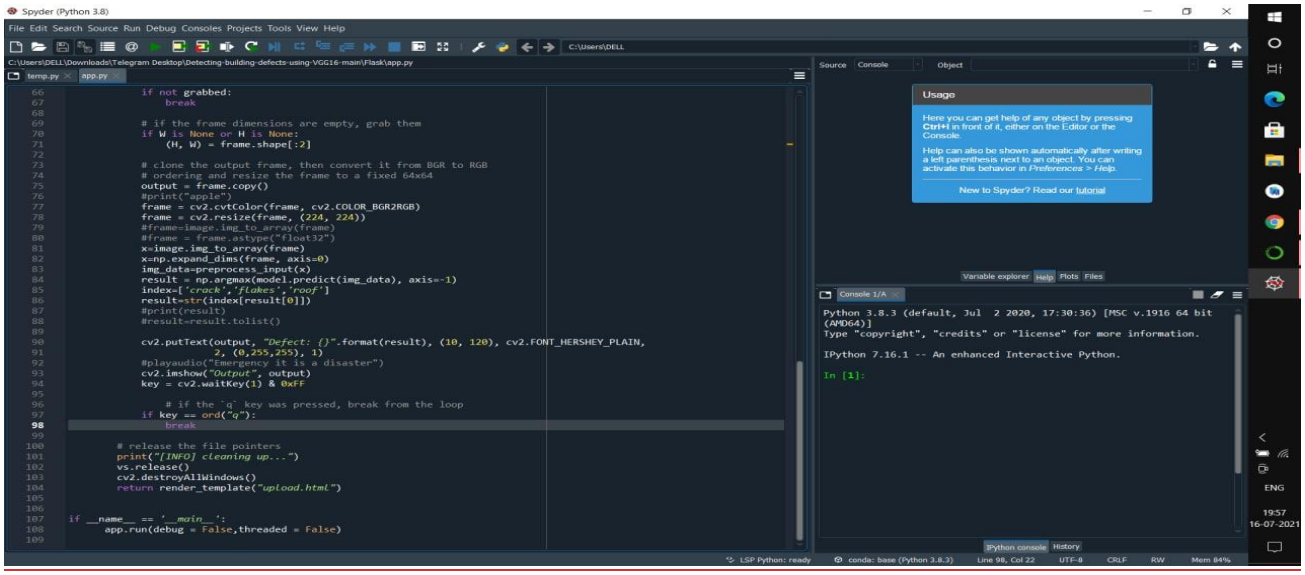
a.SOURCE CODE :



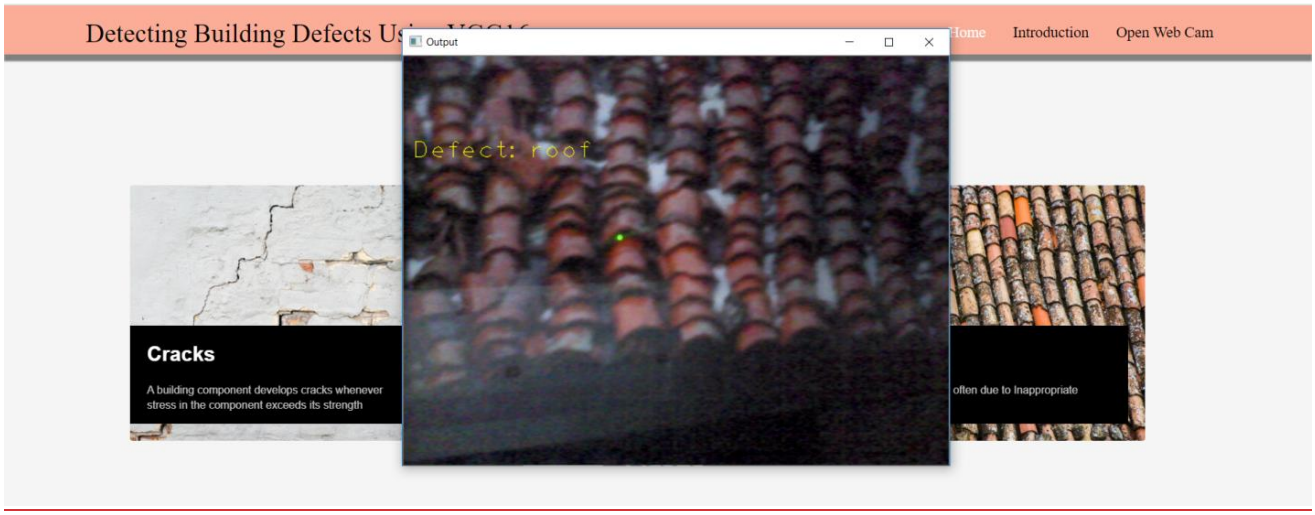
```
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 24 00:48:19 2020

@author: Tulasi
"""


# USAGE

# import the necessary packages
from flask import Flask,render_template,request
import tensorflow as tf

# 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 tensorflow.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")
print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices('GPU')))


#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=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")
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 7.16.1 -- An enhanced Interactive Python.
In [1]:

b. UI OUTPUT :


Introduction
Open Web Cam
Output
Defect: roof
Cracks
A building component develops cracks whenever stress in the component exceeds its strength
often due to inappropriate