### 

### Detecting Building Defects Using VGG16 & IBM Watson

**1.INTRODUCTION**

1.1Overview

A brief description about your project

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 significantly affect the structural integrity and the aesthetic aspect of buildings, timely and cost-effective methods of building condition surveys 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.

1.2Purpose

The use of this project. What can be achieved using this.

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.

2. **LITERATURE SURVEY**

2.1Existing problem Existing approaches or method to solve this problem

The major objective of this research has therefore is set to investigate the novel application of deep learning method of convolutional neural networks (CNN) in automating the condition assessment of buildings. The focus is to automated detection and localisation of key defects arising from dampness in buildings from images. However, as the first attempt to tackle the problem, this paper applies a number of limitations. Firstly, multiple types of the defects are not considered at once. This means that the images considered by the model belong 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, however, these limitations will be considered to be able to get closer to the concept of a fully automated detection.



### Detecting Building Defects Using VGG16 & IBM Watson

2.2Proposed solution

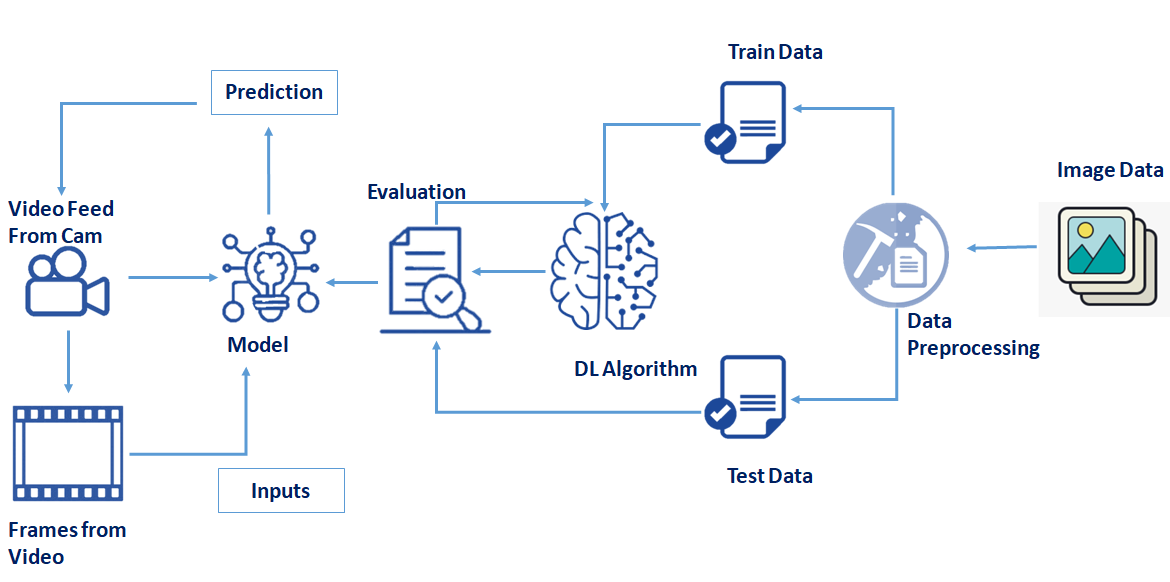
What is the method or solution suggested by you?

The aim of this research is to develop a model that classifies defects arising from dampness as “mould”, “stain” or “deterioration”, should they appear in a given image, or as “normal” otherwise. In this work we also examine the extent of ConvNets role in addressing challenges arising from the nature of the defects under investigation and the surrounding environment. For example, according to one study, mould in houses can be black, brown, green, olive-green, gray, blue, white, yellow or even pink . Moreover, stains and paint deterioration do not have a defined shape, size or colour and their physical characteristics are heavily influenced by the surrounding environment, i.e., the location (walls, ceilings, corners, etc.), the background (paint colour, wallpaper patterns, fabric, etc.) and by the intensity of light under which images of these defects were taken. The irregular nature of the defects imposes a big challenge when obtaining an adequate large-enough dataset to train a model to classify all these cases.

**3.THEORITICAL ANALYSIS**

3.1Block diagram

Diagrammatic overview of the project.





### 

### Detecting Building Defects Using VGG16 & IBM Watson

3.2Hardware / Software designing

Hardware and software requirements of the project.

1. **To build Machine learning models you must require the following packages**

* **Numpy**:
* It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations
* **Scikit-learn:**
* It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy

* **OpenCV**
* [OpenCV](https://en.wikipedia.org/wiki/OpenCV) is a library of programming functions mainly aimed at real-time computer vision. Here, OpenCV is used to capture frames by accessing the webcam in real-time.
* Open anaconda prompt and type command

“pip install opencv-contrib-python”

* **Flask:**

Web framework used for building Web applications

* **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.



### Detecting Building Defects Using VGG16 & IBM Watson

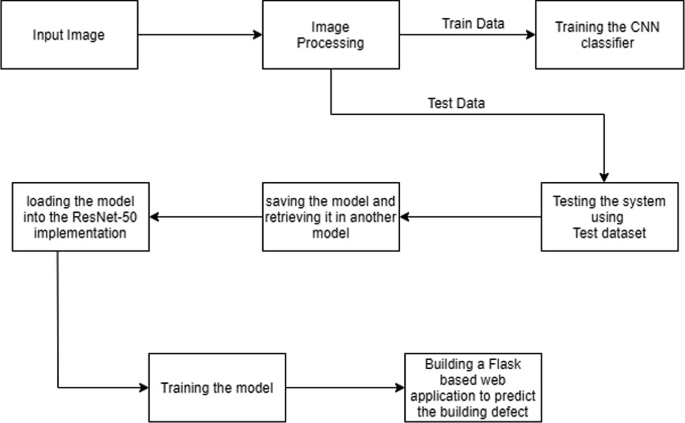
**4.EXPERIMENTAL INVESTIGATIONS**

Analysis or the investigation made while working on the solution.

The network was trained over 50 epochs using batch of size of 32 images and a step of 250 images per epoch. The final accuracy recorded at the end of the 50th epoch was 97.83% for training and 98.86% for the validation . The final loss value was 0.0572 for training and 0.042 on the validation set . The plot of accuracy in , also shows that the model has trained well although the trend for accuracy on both validation, and training datasets is still rising for the last few epochs. It also shows that all models have not over-learned the training dataset, showing similar learning skills on both datasets despite the spiky nature of the validation curve. Similar learning pattern can also be observed from  as both datasets are still converging for the last few epochs with a comparable performance on both training and validation datasets.  also shows that all models had no overfitting problem during the training as the validation curve is converging adjacently to training curve and has not diverted away from the training curve.

**5.FLOWCHART**

Diagram showing the control flow of the solution



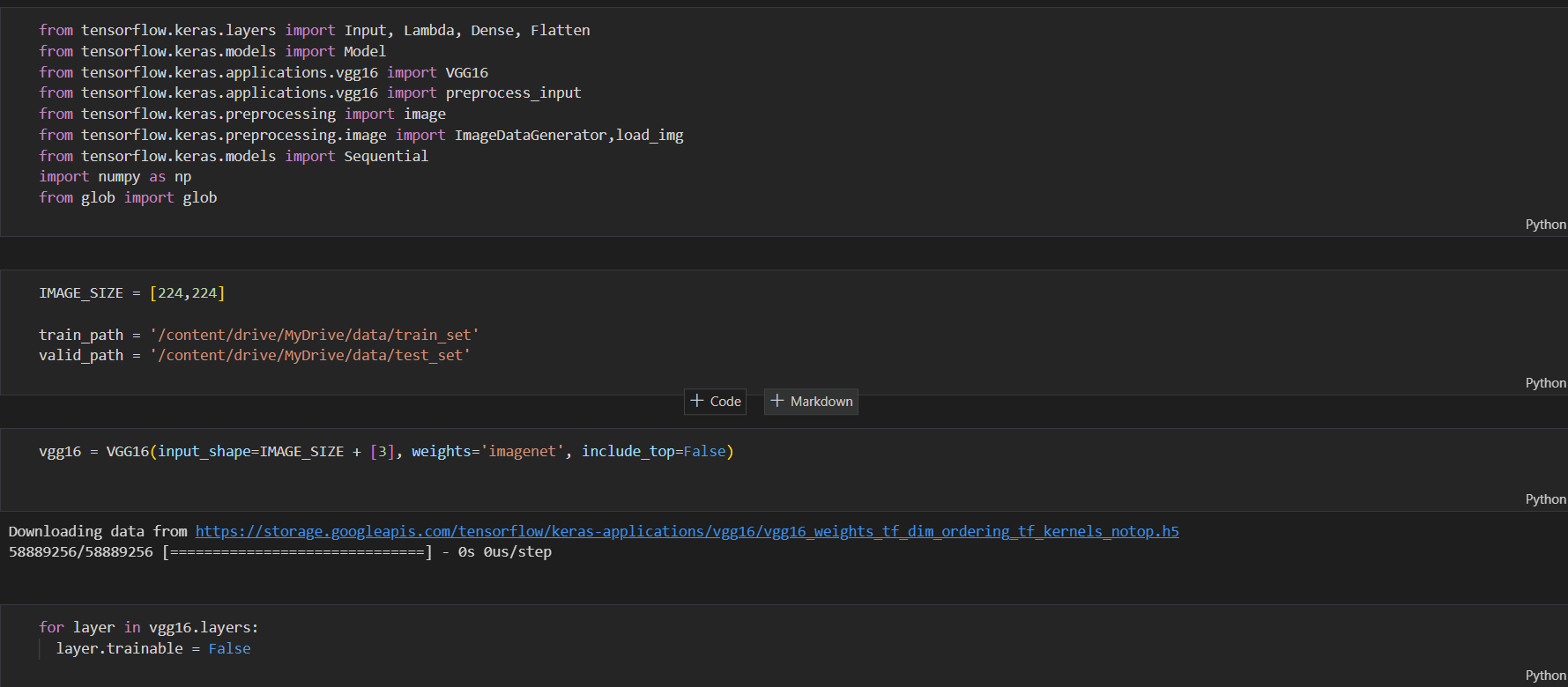


### Detecting Building Defects Using VGG16 & IBM Watson

**6.RESULT**

Final findings (Output) of the project along with screenshots.

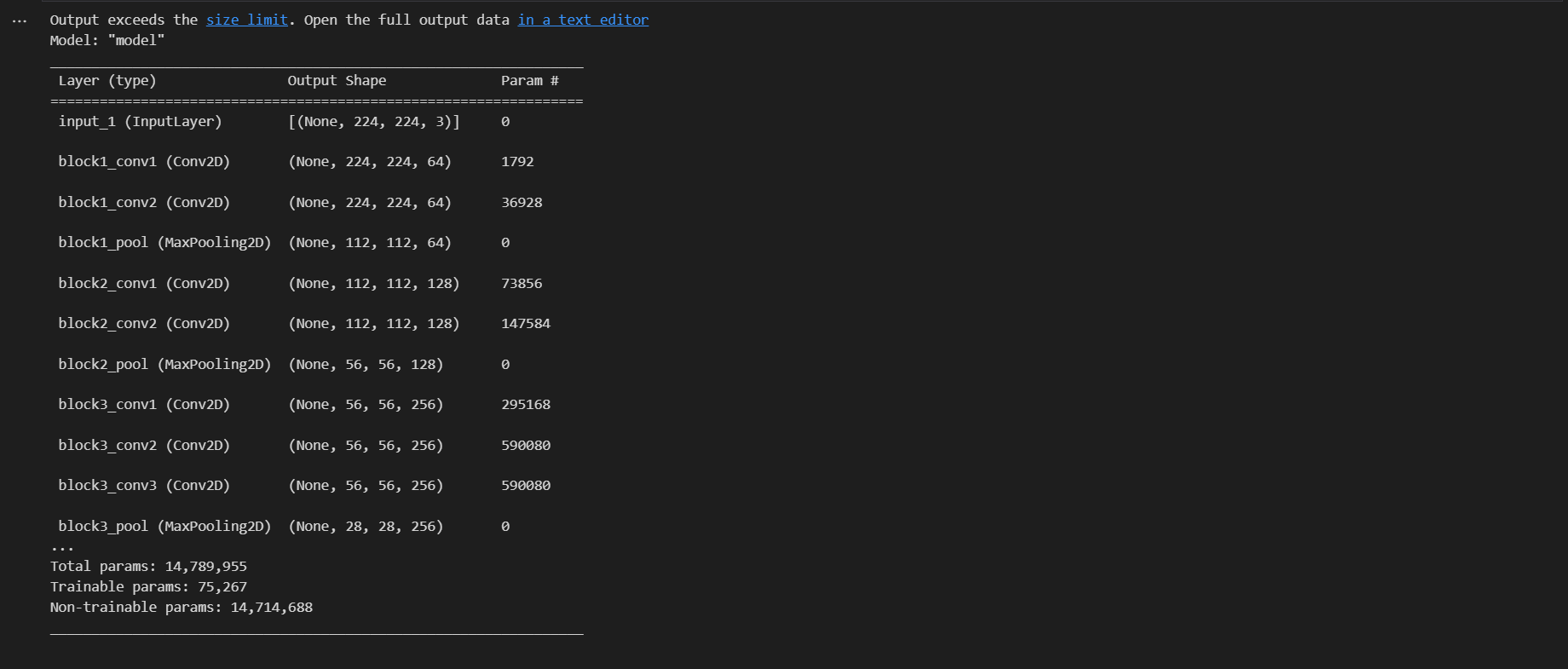


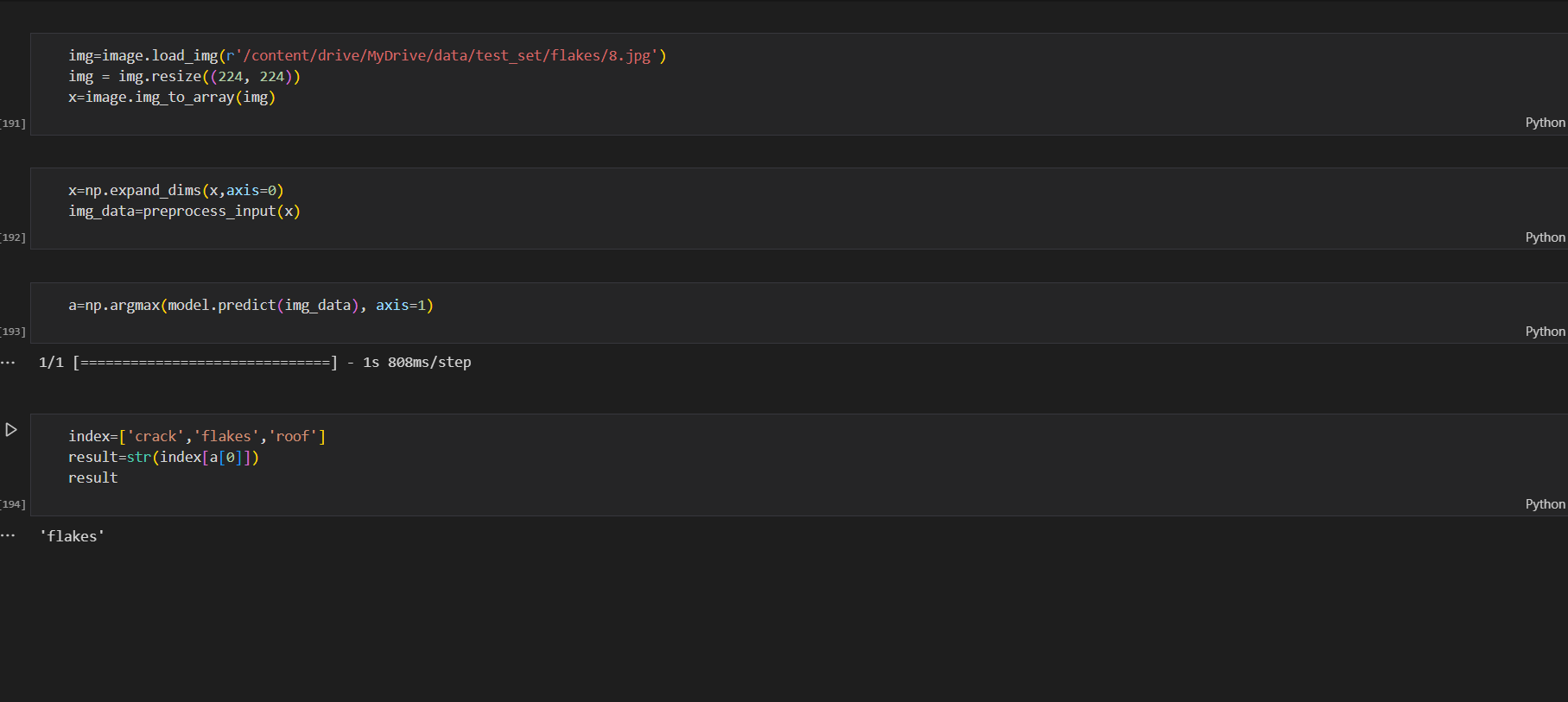
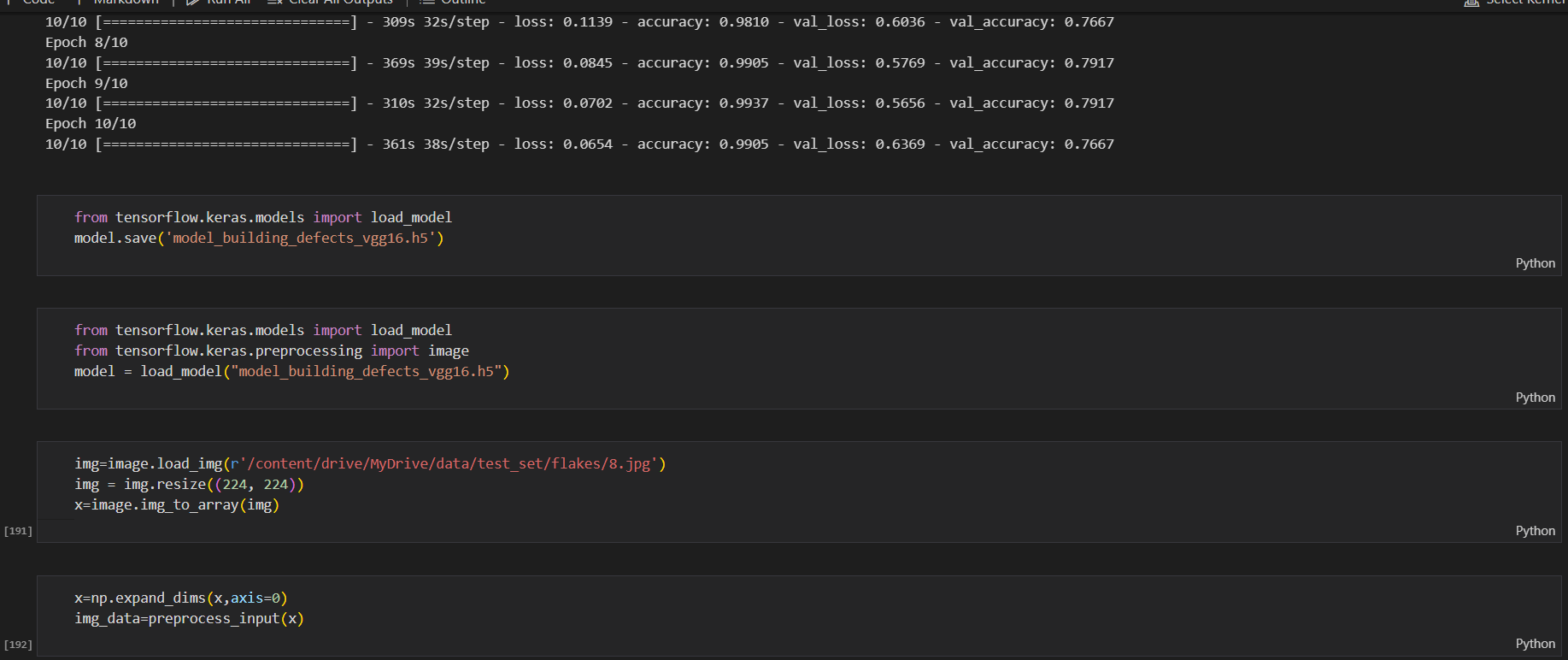
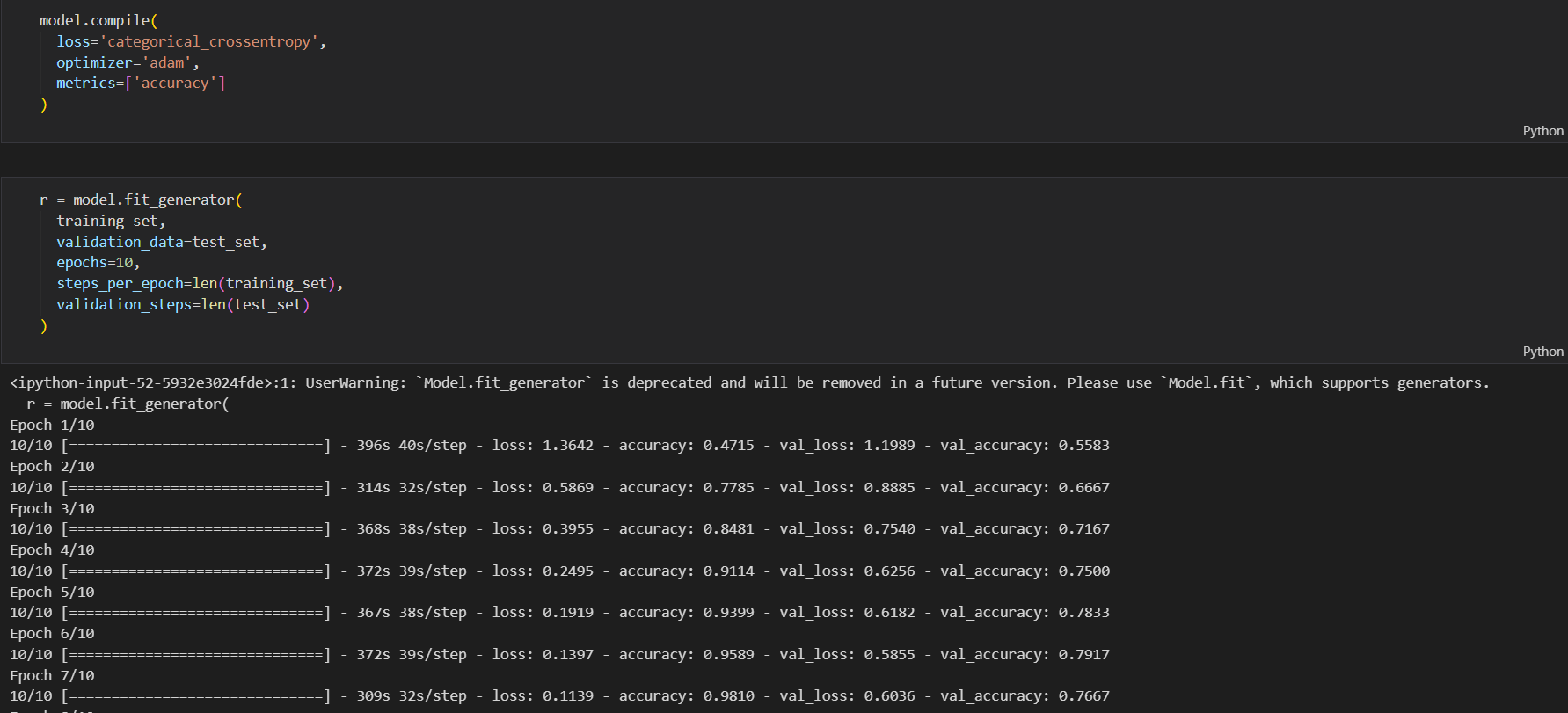


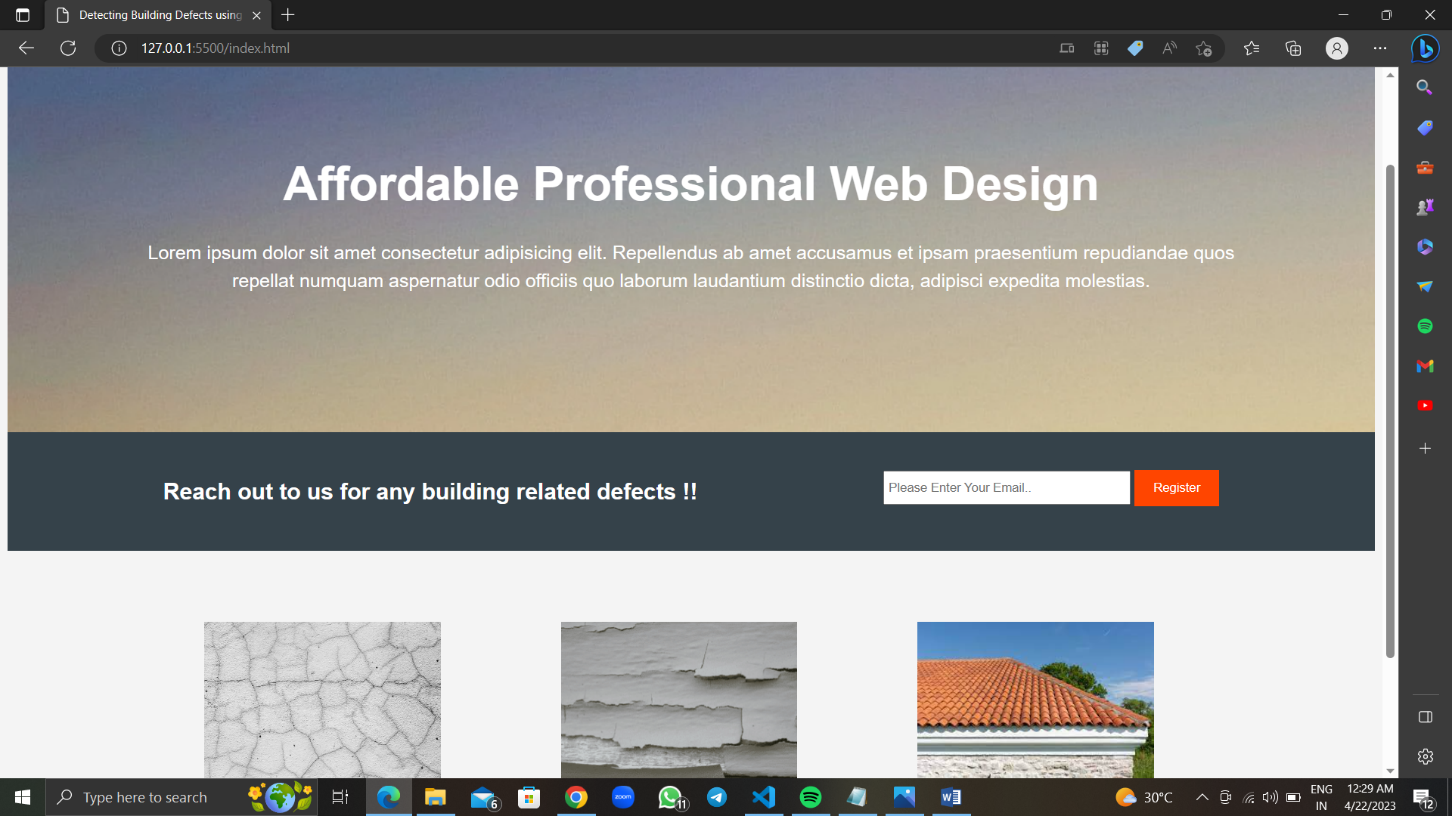


### Detecting Building Defects Using VGG16 & IBM Watson











### Detecting Building Defects Using VGG16 & IBM Watson

7.**ADVANTAGES & DISADVANTAGES**

List of advantages and disadvantages of the proposed solution

Advantages:

* CNNs do not require human supervision for the task of identifying important features.
* They are very accurate at image recognition and classification.
* Weight sharing is another major advantage of CNNs.
* Convolutional neural networks also minimize computation in comparison with a regular
* .0neural network.
* CNNs make use of the same knowledge across all image locations.

Disadvantages:

Classification of Images with different Positions

Adversarial examples

Coordinate Frame

Other minor disadvantages like performance

**8.APPLICATIONS**

The areas where this solution can be applied

For the purpose of this research, images containing the defect types were collected from different sources. The images were then appropriately, cropped and resized to generate the dataset which was used to train our model. To achieve higher accuracy, instead of training a model from scratch, we adopted a transfer learning technique and used a pre-trained VGG-16 on ImageNet as our chosen model to customise and initialise weights. A separate set of images, not seen by the trained model, was used for validation to examine the robustness of our model.



### Detecting Building Defects Using VGG16 & IBM Watson

**9.CONCLUSION**

Conclusion summarizing the entire work and findings.

* know fundamental concepts and Transfer Learning techniques.
* gain a broad understanding of image data.
* Know how to pre-process/re-size the data using different data preprocessing techniques.
* know how to build a web application using the Flask framework.

**10.FUTURE SCOPE**

Enhancements that can be made in the future.

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.

**11.BIBILOGRAPHY**

References of previous works or websites visited/books referred foranalysis about the project, solution previous findings etc.

<https://github.com/Guided-Projects/AI-based-Natural-disaster-analysis>

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6720984/>

[https://www.mdpi.com/1424-8220/19/16/3556#](https://www.mdpi.com/1424-8220/19/16/3556)



### Detecting Building Defects Using VGG16 & IBM Watson

**APPENDIX**

Source Code:-

Attach the code for the solution built.

from keras.preprocessing.image import ImageDataGenerator

#performing data argumentation on train data

train\_datagen = ImageDataGenerator(rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizantal\_flip = True)

#performing data argumentation on test data

test\_datagen = ImageDataGenerator(rescale = 1./255)

#make sure you provide the same target size as initialized for the image size

training\_set = train\_datagen.flow\_from\_directory('../data/train\_set',

target\_size = (224, 224),

batch\_size = 32,

class\_mode = 'categorical')

test\_set = test\_datagen.flow\_from\_directory('../dat/test\_set',

target\_size = (224, 224),

batch\_size = 32,

class\_mode = 'categorical')

import the libraries as shown below

from tensarflow.keras.layers import Input, Lambda, Dense, flatten

from tensarflow.keras.models import Model

from tensarflow.keras.applications.vgg16 import VGG16

from tensarflow.keras.applications.vgg16 import preprocess\_input

from tensarflow.keras.preprocessing import image

from tensarflow.keras.preprocessing.image import ImageDataGenerator,load\_img

from tensarflow.keras.models import Sequential

import numpy as np



### Detecting Building Defects Using VGG16 & IBM Watson

from glob import glob

IMAGE\_SIZE = [224,224]

train\_path = 'data/train\_set'

valid\_path = 'data/test\_set'

# Import the vgg16 library as shown below and preprocessing layer to the front of vgg

# Here we will be using imagenet weights

vgg16 = VGG16(input\_shape=IMAGE\_SIZE + [3], weights='imagenet', include\_top=False)

# don't train existing weights

for layer in vgg16.layers:

layer.trainable = False

# useful for getting number of output classes

folders = glob('../data/train\_set/\*')

# our layers - you can add more if you want

x = flatten()(vgg16.output)

prediction = Dense(len(folders), activation='softmax')(x)

# create a model object

model = Model(inputs=vgg16.input, outputs=prediction)

# view the structure of the model

model.summary()

COMPILING THE MODEL

# tell the model what cost and optimization method to use

model.compile(

loss='categorical\_crossentropy',

optimizer='adam',

metrics=['accuracy']

)

FIT THE MODEL

# fit the model

# run the cell. It will take some time tto execute

r = model.fit\_generator(

training\_set,



### Detecting Building Defects Using VGG16 & IBM Watson

validition\_data=test\_set,

epochs=10,

steps\_per\_epoch=len(training\_set),

validation\_steps=len(test\_set)

)

SAVE THE MODEL

from tensarflow.keras.models import load\_model

model.save('model\_building\_defects\_vgg16.h5')

TEST THE MODEL

from tensarflow.keras.models import load\_model

fromkeras.preprocessing.image import image

model = load\_model("model\_building\_defects\_vgg16.h5") #loading the model for testing

img=image.load\_imp(r'D:\ML\_training may 2020\projects\_50\Final\Deducting\_building\_defects\data\test\_set\flal

x=imahe.img\_to\_array(img)

#X=x/255

x=np.expand\_dims(x,axis=0)

img\_data=preprocess\_input(x)

#model.predict(img\_data)

a=np.argmax(model.predict(img\_data), axis=1)

index=['crack','flakes','roof']

result=str(index[a[0]])

result