Guided Project

Detecting Building Defects Using VGG16 & IBM Watson

~ By The Datamatics

Kithuru Fathima | Safiya Fathima | Aishwarya R Nair | Diva Lade

INDEX

S.NO	Topic	Page No
1.	Abstract	3
2.	Introduction	3
3.	Theory	4
4.	Model building and implementation in local system	7
5.	Integration with Flask	8
6.	Model building in IBM Watson Studio	11
7.	References	12

ABSTRACT

An efficient method to observe and convey the condition of building so that important repairs and maintenance can be performed in a proactive way is growing in demand by clients swiftly and recurrently. Such a method can help in preventing repairs before they become dangerous or expensive. The traditional methods for such a task are time consuming, labor-extensive, expensive and pose heath issues. It includes 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. We aim to involve the concept of convolution neural networks (CNN) to automate the process of surveying buildings, specifically for defects such as cracks, flakes and roof flaws. This model is built on the concept of CNN and uses VGG-16 and OpenCV. The model is robust and can accurately detect building defects in real time.

INTRODUCTION

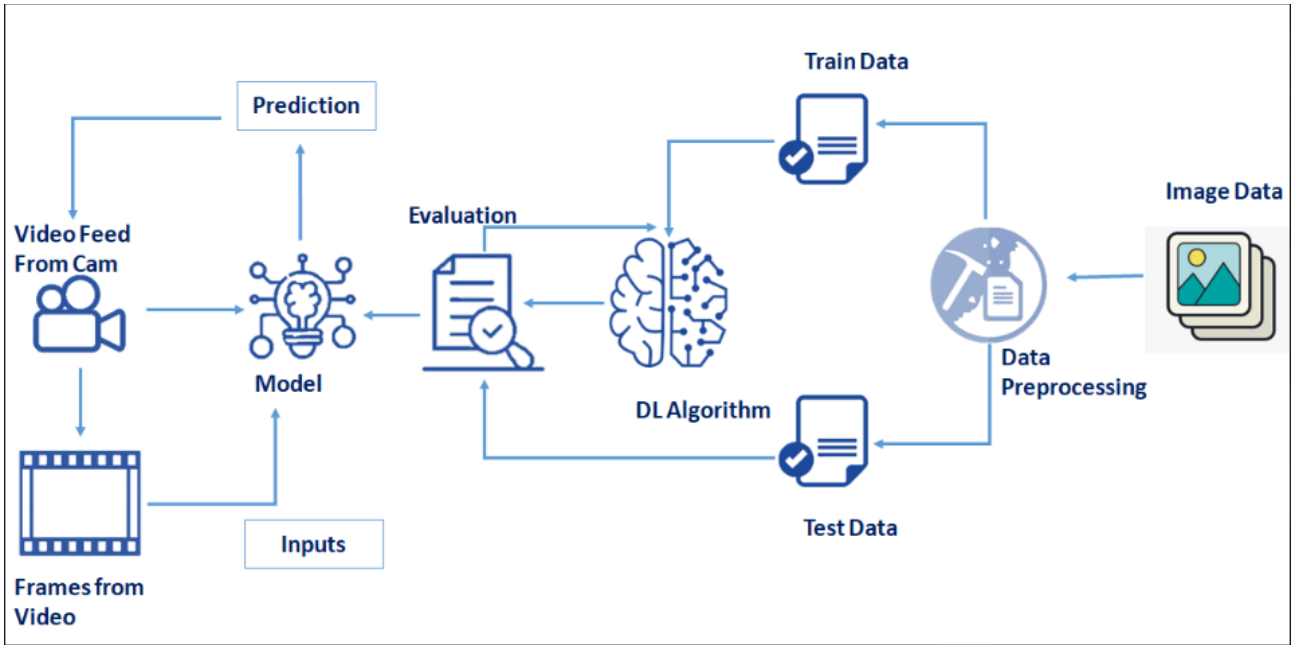
Detection of building defects including cracks and flakes on the wall surfaces is a crucial task of buildings' maintenance, which if left undetected and untreated, can significantly affect the structural integrity and the aesthetic aspect of buildings.

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 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.

The current asset condition testing/inspecting techniques are time-consuming, labour-intensive, demanding, as well as posing health and safety risks to surveyors, particularly at heights and on rooftops that are difficult to inspect. As an alternative to manual on-site inspection methods, image analysis algorithms for detecting defects have been proposed. Whereas the latter is time-consuming and unsuitable for quantitative analysis, image analysis-based detection techniques, on the other hand, can be challenging and completely reliant on the quality of images captured in various real-world scenarios.

In this project to detect defects such as cracks , flakes, and roof defects, we used CNN pre-trained model VGG16 to analyse 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.

Technical Architecture



THEORY

1.Convolutional Neural Networks (ConvNet)

Convolution Neural Networks, known as CNN, are a subset of machine learning and are the heart of deep learning algorithms. CNN is commonly used in classification and computer vision tasks.

CNN is comprised of node layers, which contains an input layer, one or more hidden layers, and an output layer. The more the hidden layers, the deeper is the neural network. The number of nodes in the input layer depend on the number of features in the input data, the number of nodes in the output layer, in this case, will depend upon the number of classes in the network. Here, it would have 3 nodes, corresponding to the three class labels : cracks, flakes, and roof. Each node is connected to another and is associated with a weight and threshold.

The convolution layer is the core and most important block of CNN. The majority of computation occurs here and it requires the input data, a filter and a feature map. The input data, which is a colour image, is expressed as a 3-D array. A feature detector, known as kernel or filter, moves across the fields of the image to check if a feature is present. This method is known as convolution.

The pooling layer reduces the number of parameters in the input. It moves across the input but it does not have any weights. It breaks down the output of the convolution layers into smaller regions and the maximum value of every smaller region is taken out. This is called Max-Pooling.

Fully Connected layers allow each node in the output layer to connect directly to a node in the previous layer. The layer performs the classification of features extracted through the previous layers.

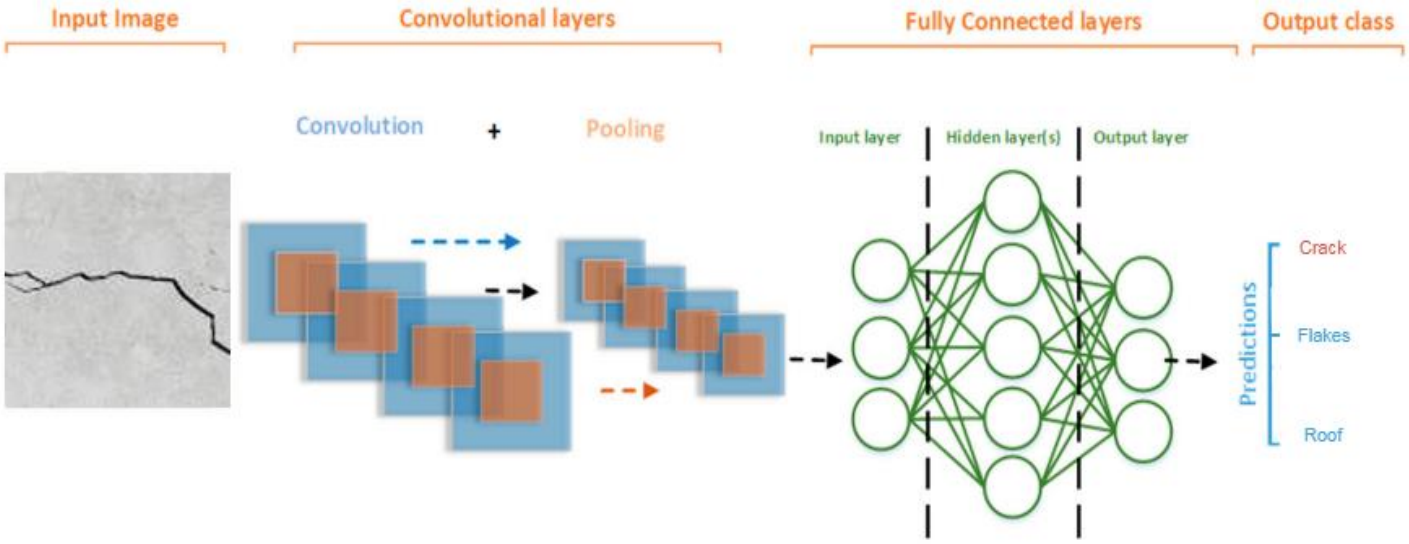


Fig: ConvNet Architecture

2.VGG16

The VGG-16 is proven to be a powerful network although having a simple architecture. This simplicity makes it easier to modify for transfer learning and for the CAM technique without compromising the accuracy. Moreover, VGG-16 has fewer layers (shallower) than other models such as the ResNet50 or Inception. Since the VGG-16 has fewer layers than other networks, it makes it a better model to train on our relatively small dataset compared to deeper neural networks showing, accuracy are close, however VGG-16 training is smoother.

The architecture of the VGG-16 model comprises five blocks of convolutional layers with max-pooling for feature extraction. The convolutional blocks are followed by three fully connected layers and a final 1×1000 Softmax layer (classifier). The input to the ConvNet is a3-channel (RGB) image with a fixed size of 224 × 224. The first block consists of two convolutional layers with 32 filters, each of size 3×3. The second, third, and forth convolution blocks use filters of sizes 64×64×3, 128×128×3, and 256×256×3, respectively.

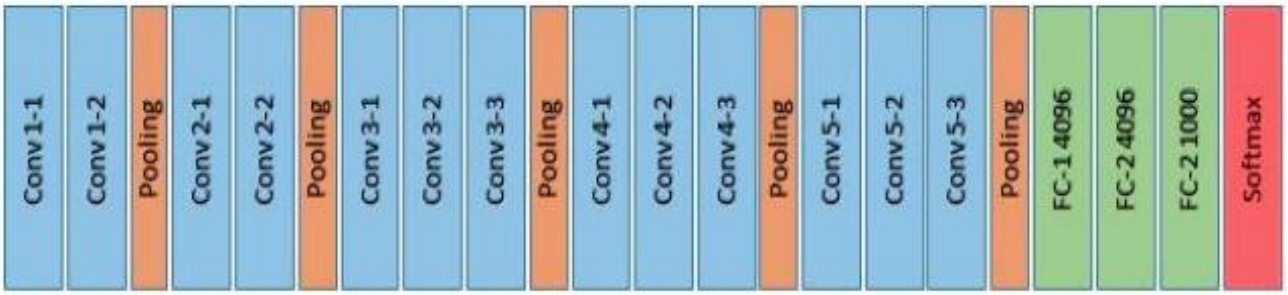


Fig: VGG-16 architecture.

Tools Used:

NumPy

Numpy is an open-source numerical Python library, that offers array and matrix data structures and mathematical operations and more.

Scikit-learn

Scikit-learn is a free software machine learning Python library, offering tools for predictive data analysis like classification, regression and clustering algorithms, linear algebra and so on.

VGG-16

VGG-16 is a convolution neural network (CNN) architecture and is an excellent vision model architecture. It is used to extract features and classify images.

OpenCV

OpenCV is an open-source computer vision and machine learning software library. It includes more than 2500 optimized algorithms that can detect and recognize faces, identify objects, track camera movements and so on.

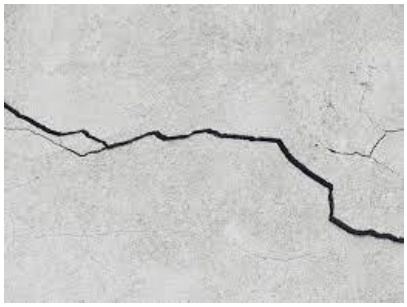
Flask

Flask is a a lightweight WSGI web application framework. It has less base code to implement a simple web-Application and also provides the ability to scale up to more complex applications.

Dataset

The dataset generated is for Building Defects Model. The training data is separated into 3 classes:(crack,flake,roof) but the test data is gathered. The dataset has 1,164 texture images(formats:png,jpeg,jpg) and 2 cateogies : Defective and Normal

Training and Test data:
The training data is randomly chosen (70%) instances for each class, the total number of training data images is 816.
The test data is the remaining (30%) instances for each class, the total number of test data images is 318.
Both training and test data is separated into 3 classes.



i) crack

ii) flakes

iii) roof

Fig: Three types of building defects

MODEL BUILDING AND IMPLEMENTATION IN LOCAL SYSTEM

1. Image Pre-processing

1.1. Import the required libraries

from keras.preprocessing.image import ImageDataGenerator

ImageDataGenerator library is used to implement image data augmentation technique. It expands the training dataset to improve the performance and ability of the model and generalizes it.

An instance of the ImageDataGenerator class is constructed for train and test and apply ImageDataGenerator functionality to Trainset and Testset.

2. Model Building

For our model, we applied a fine-tuning transfer learning to a VGG-16 network. It is a pre trained CNN model on a huge dataset (ImageNet) with a lot of diverse image categories.

2.1. Import the required libraries

```
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
```

2.2. Data cleaning

The images are resized to standardize the size of the images in the dataset.

2.3. Creation of the CNN Model

The pre-trained CNN Model is used as a feature extractor.

Dense layer is added. An object of the model is created, and input is given as VGG16.input and output as dense layer. Activation function used is softmax.

2.3.1. Compiling the model

It is the final step in creating a model. We have used loss function in Keras during the model compilation process. The adam optimizer is used to optimize the input weights by comparing the prediction and the loss function. Metrics used is accuracy.

2.4. Train the model

We trained the model with our image dataset. The model is trained for 10 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time.

2.5. Save the model

The generated model is saved as a file with .h5 extension.

2.6. Test the model

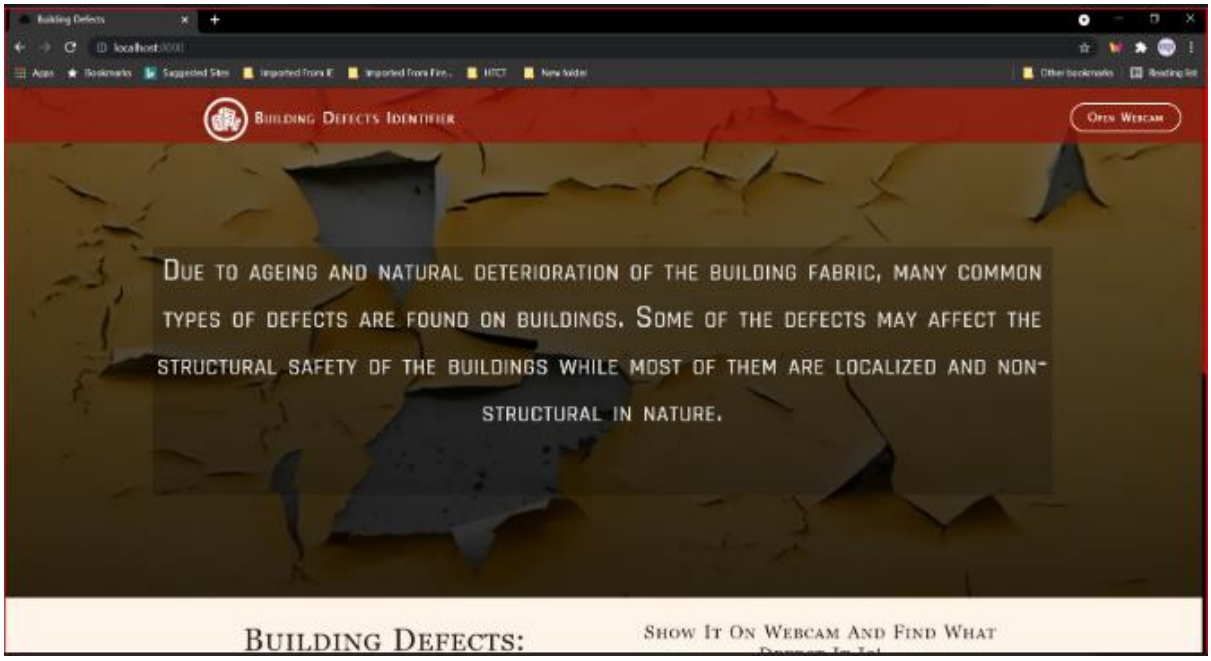
We tested the model to check if it's the best fit for the given problem and corresponding data.

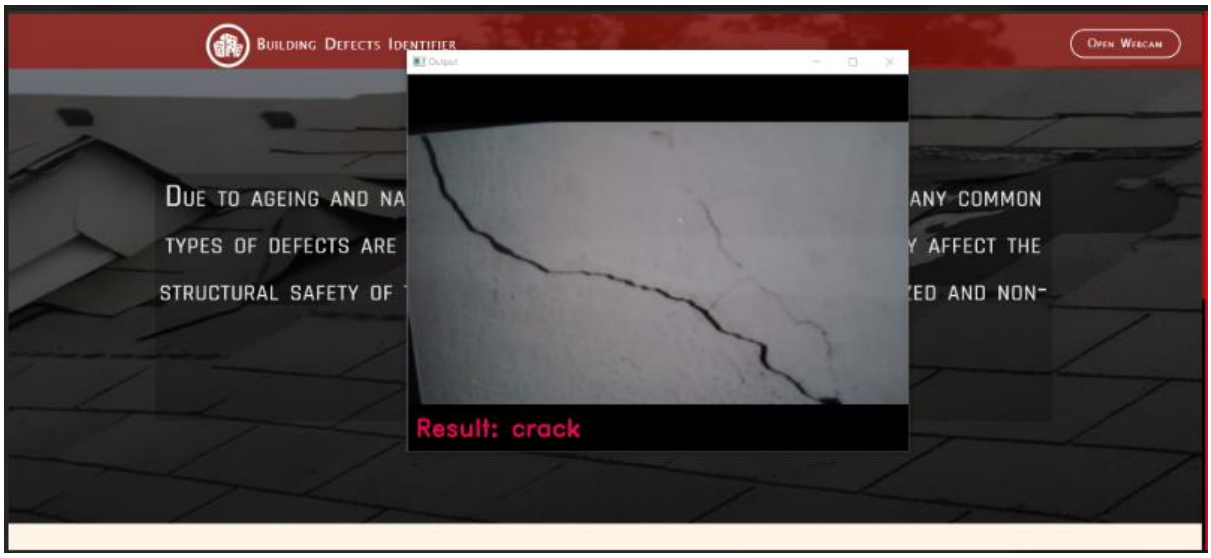
Load the saved model using *load model*. Take an image as input and check the results. Using the model, we predicted the output for the given input image.

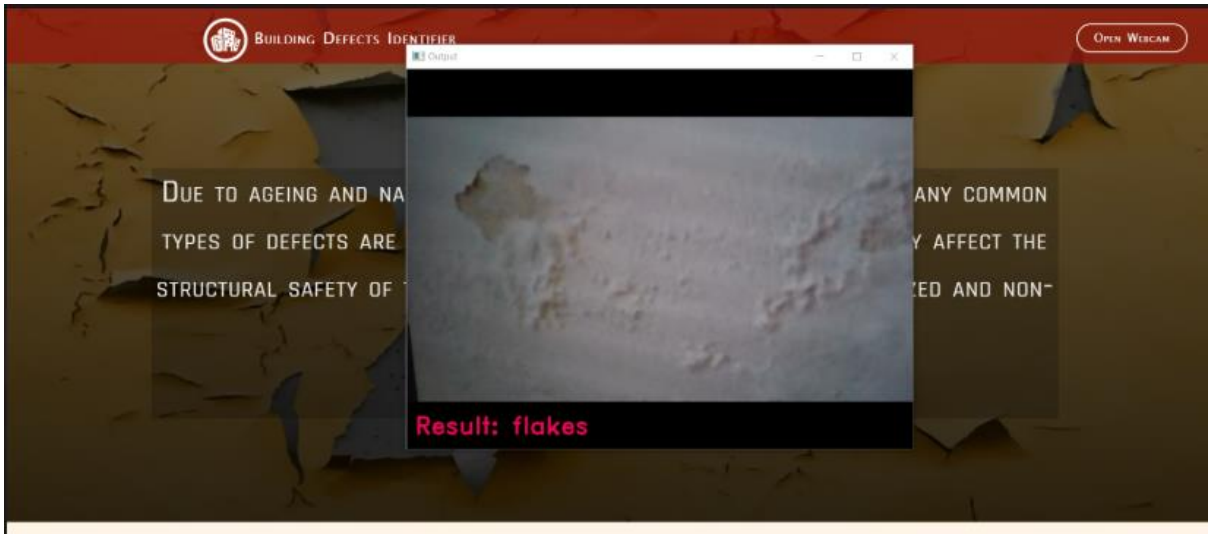
INTEGRATION WITH FLASK

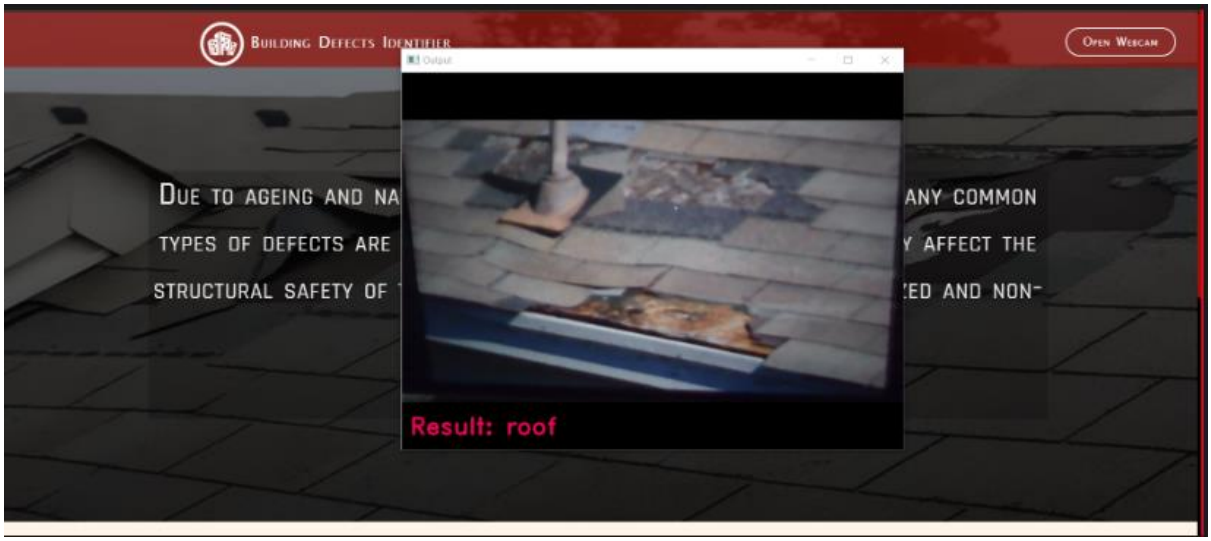
Required libraries related to open cv and VGG16 have been imported in the python code. Our flask application has two routes: '/' (leads to home.html) and '/end' (leads to end.html). We have used CSS for styling the page. The webcam opens when the

‘Open Webcam’ button is clicked. Objects are shown through the webcam and our CNN model predicts the type of defect and displays it on the flask webpage.




Building Defects Identifier
Open Webcam
Output
Result: crack


Building Defects Identifier
Open Webcam
Output
Result: flakes


Building Defects Identifier
Open Webcam
Output
Result: roof

MODEL BUILDING IN IBM WATSON STUDIO

Libraries used:-

1) keras:for developing and evaluating deep learning models.

class imported: ImageDataGenerator

2) tensforflow: for fast numerical computing created.

class imported:Input, Lambda, Dense, Flatten,

Model, VGG16, preprocess_input, image, load_img, Sequential

3) BytesIO: methods that manipulate bytes data in memory.

class imported: io, zipfile

4) glob: used to return all file paths that match a specific pattern.

5) numpy: to perform a wide variety of mathematical operations on arrays.

6) pandas:is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool.

7) os:provides functions for interacting with the operating system.

8) types: contains type objects for all object types defined by the standard interpreter.

9) botocore.client:low-level interface to a growing number of Amazon Web Services.

class imported: Config

10) ibm_boto3:the IBM COS SDK for Python, which allows Python developers to write software that makes use of IBM's COS service.

REFERENCES

- https://www.ibm.com/cloud/learn/convolutional-neural-networks
- https://www.academia.edu/40089851/Deep_Learning_for_Detecting_Building_Defects_Using_Convolutional_Neural_Networks