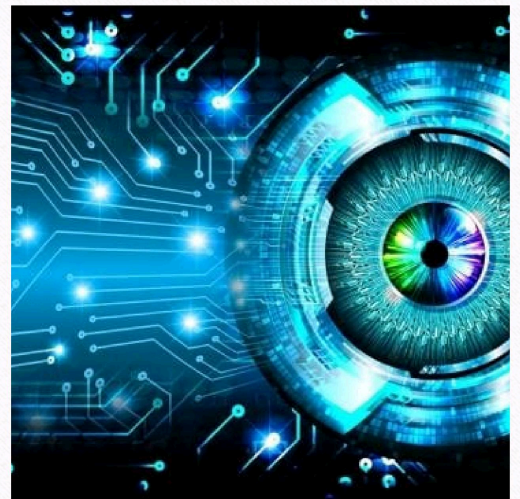


IMAGE RECOGNITION WITH IBM CLOUD VISUAL

- INTRODUCTION
- THE PROBLEM
- IBM WATSON
- HOW IT WORKS
- PRICING
- CONTEXT
- CONCLUSION

INTRODUCTION

- **IBM Watson** Visual recognition is a powerful tool that can enhance image analysis workflow.
- This cloud based service uses deep learning algorithms to identify objects, faces and scenes in image.



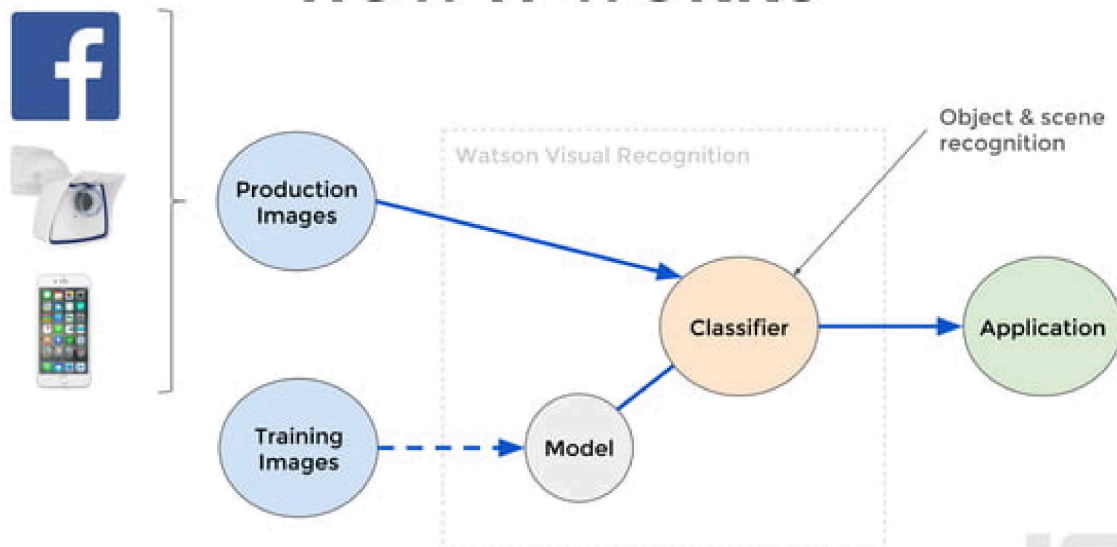
THE PROBLEM:

- We live in a visual world yet capturing useful information from images has historically required human vision which can be slow and costly.

IBM WATSON



HOW IT WORKS



10x

PRICING

Free tier

- 250 images/day + one custom classifier (train 1000 images)

Production

- Image Tagging – \$0.002/image
- Face Detection – \$0.004/image
- Training – \$0.25 USD/Event
- Custom Tagging – \$0.004/event
- Custom Model Storage – \$10.00/mo

10x

CONTEXT

Works for all businesses.

Digital



Physical

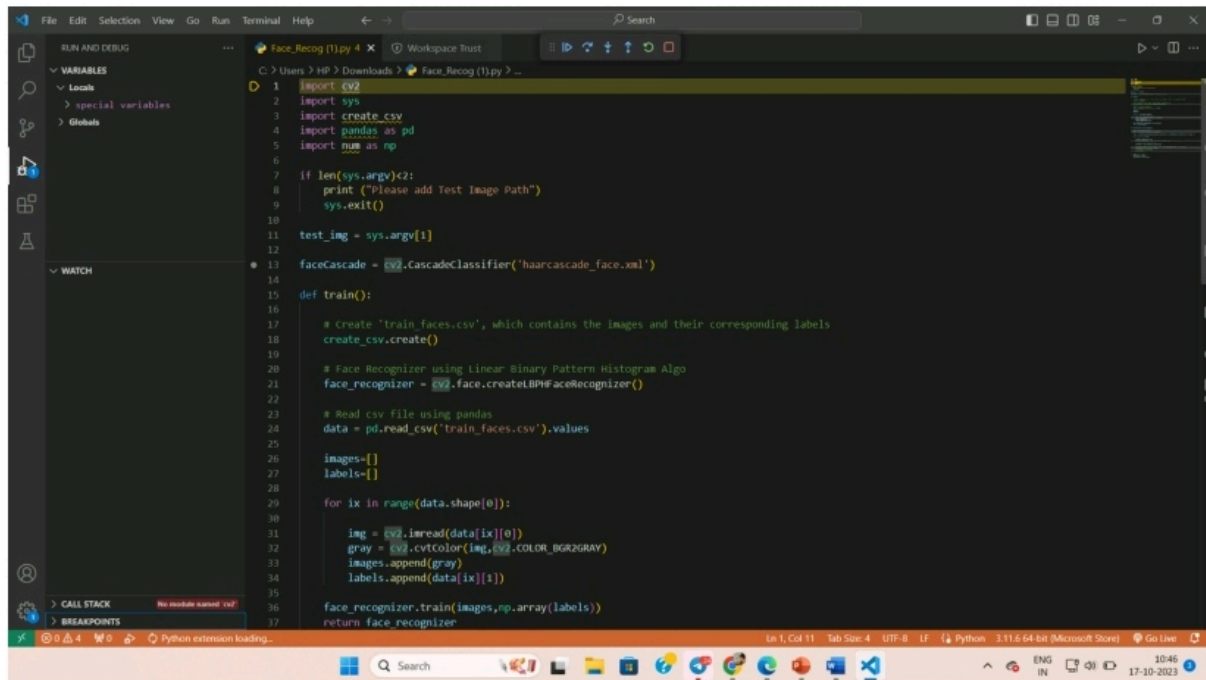


CONCLUSION :

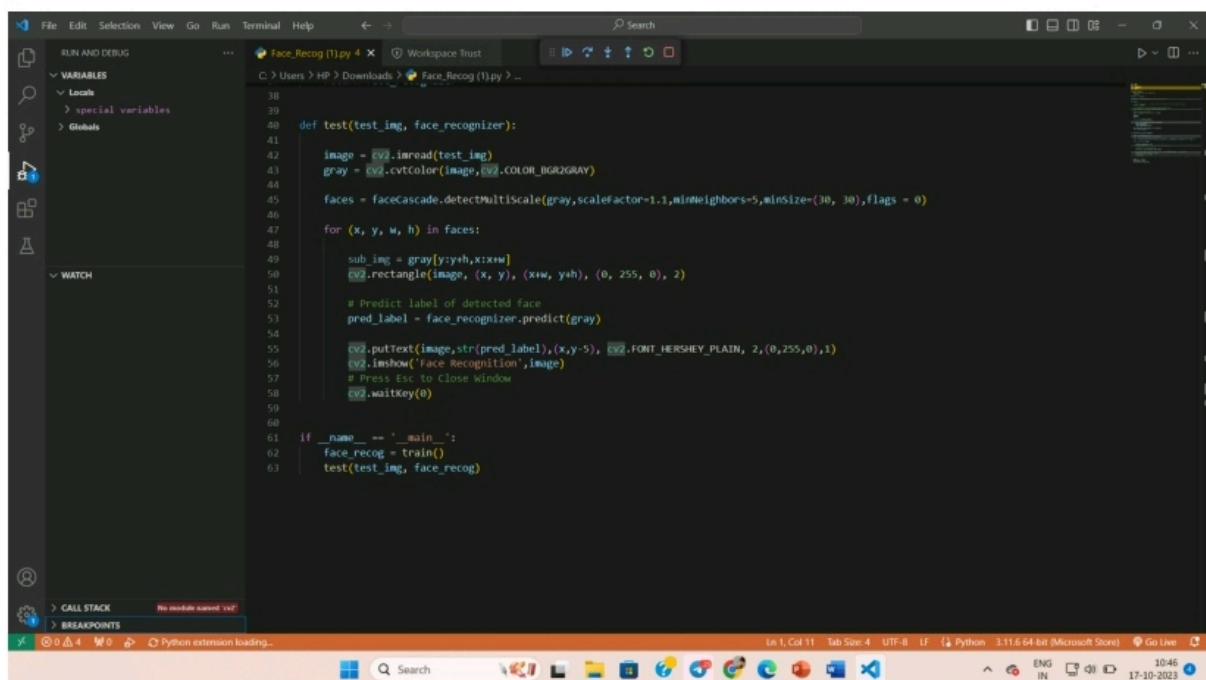
Image recognition technology has transformed the way we process and analyze digital images and videos making it possible to identify objects, diagnose diseases and automate workflows accurately and efficiently.

Image Recognition with IBM Cloud Visual Recognition

Phase 3: Development Part 1



```
1 import cv2
2 import sys
3 import create_csv
4 import pandas as pd
5 import argparse
6
7 if len(sys.argv) < 2:
8     print("Please add Test Image Path")
9     sys.exit()
10
11 test_img = sys.argv[1]
12
13 faceCascade = cv2.CascadeClassifier('haarcascade_face.xml')
14
15 def train():
16     # Create 'train_faces.csv', which contains the images and their corresponding labels
17     create_csv.create()
18
19     # Face Recognizer using Linear Binary Pattern Histogram Algo
20     face_recognizer = cv2.face.LBPHFaceRecognizer_create()
21
22     # Read csv file using pandas
23     data = pd.read_csv('train_faces.csv').values
24
25     images = []
26     labels = []
27
28     for ix in range(data.shape[0]):
29
30         img = cv2.imread(data[ix][0])
31         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
32         images.append(gray)
33         labels.append(data[ix][1])
34
35     face_recognizer.train(images, np.array(labels))
36
37     return face_recognizer
```



```
38
39
40 def test(test_img, face_recognizer):
41
42     image = cv2.imread(test_img)
43     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
44
45     faces = faceCascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30), flags = 0)
46
47     for (x, y, w, h) in faces:
48
49         sub_img = gray[y:y+h, x:x+w]
50         cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
51
52         # Predict label of detected face
53         pred_label = face_recognizer.predict(gray)
54
55         cv2.putText(image, str(pred_label), (x, y-5), cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 0), 1)
56         cv2.imshow('Face Recognition', image)
57         # Press Esc to Close window
58         cv2.waitKey(0)
59
60
61 if __name__ == '__main__':
62     face_recog = train()
63     test(test_img, face_recog)
```