IMAGE RECOGNITION WITH IBM CLOUD VISUAL RECOGNITION

INTRODUCTION

Image recognition, also known as computer vision, is a technology that enables computers to interpret and understand visual information from images or videos. IBM Cloud Visual Recognition is a cloud-based service offered by IBM that makes it easier to build and deploy image recognition capabilities into your applications. In this introduction, we'll explore the key concepts and features of IBM Cloud Visual Recognition.

PROBLEM STATEMENT:

In today's digital age, the vast amount of visual data in the form of images and videos has become a valuable resource across industries such as healthcare, e-commerce, automotive, and security. However, the manual analysis of this data is time-consuming, error-prone, and often impractical given the scale of available information.

To harness the potential of visual data, there is a need for an automated image recognition system that can accurately and efficiently classify, detect, and analyse objects, scenes, and patterns within images

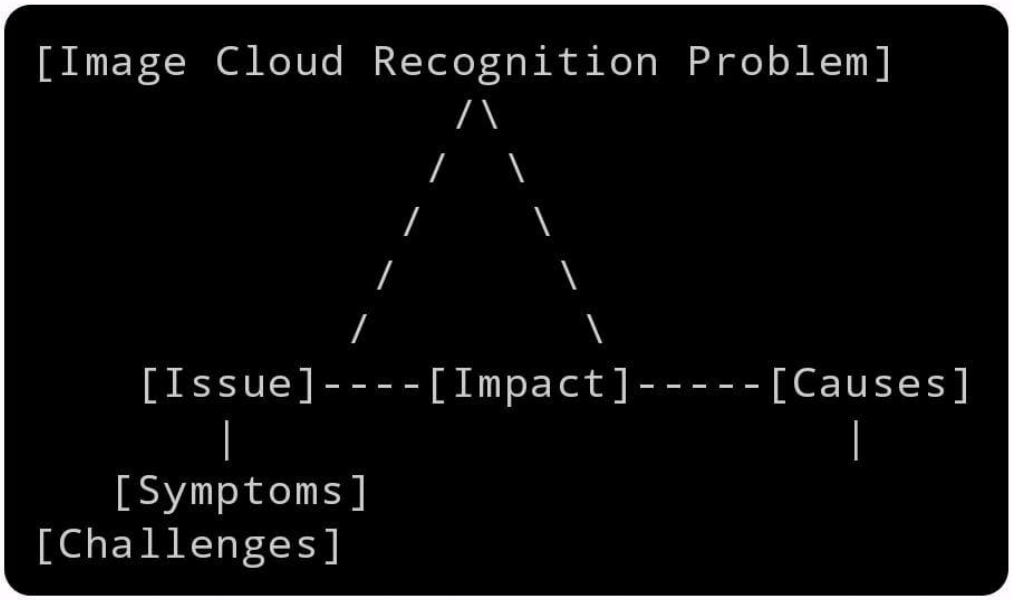


fig: Problem Statement

SYMPTOMS:

Here are some common symptoms or challenges you might encounter:

1. **Low Accuracy:** The recognition results are inaccurate or inconsistent, failing to correctly identify objects, people, or text within images.

2. **Slow Processing:** The service takes a long time to analyse or process images, which can be problematic for real-time or high-throughput applications.
3. **Limited Object Recognition:** The service may struggle to recognize objects or entities in certain images, particularly if they are unusual or not well-represented in the training data.
4. **Security and Privacy Concerns:** If working with sensitive images, there might be concerns about the privacy and security of the data being processed.
5. **Cost Overruns:** If you're not careful with your usage, you might exceed your budget due to the cost associated with processing a large number of images.
6. **Integration Challenges:** Integrating the visual recognition service into your application or workflow might pose challenges, especially if you're working with various APIs or SDKs.
7. **Complexity of Use:** Depending on the service, it might require a steep learning curve to use effectively, especially if you're not familiar with machine learning concepts.
8. **Lack of Customization:** You may want to fine-tune the recognition model for specific use cases, but some services have limitations on customization.
9. **Unsupported Image Types:** Some services may have difficulty processing certain types of images, such as low-resolution images or images with unusual formats.
10. **Data Quality Issues:** Poor-quality input images (e.g., low resolution, noisy, or distorted) can lead to inaccurate results.

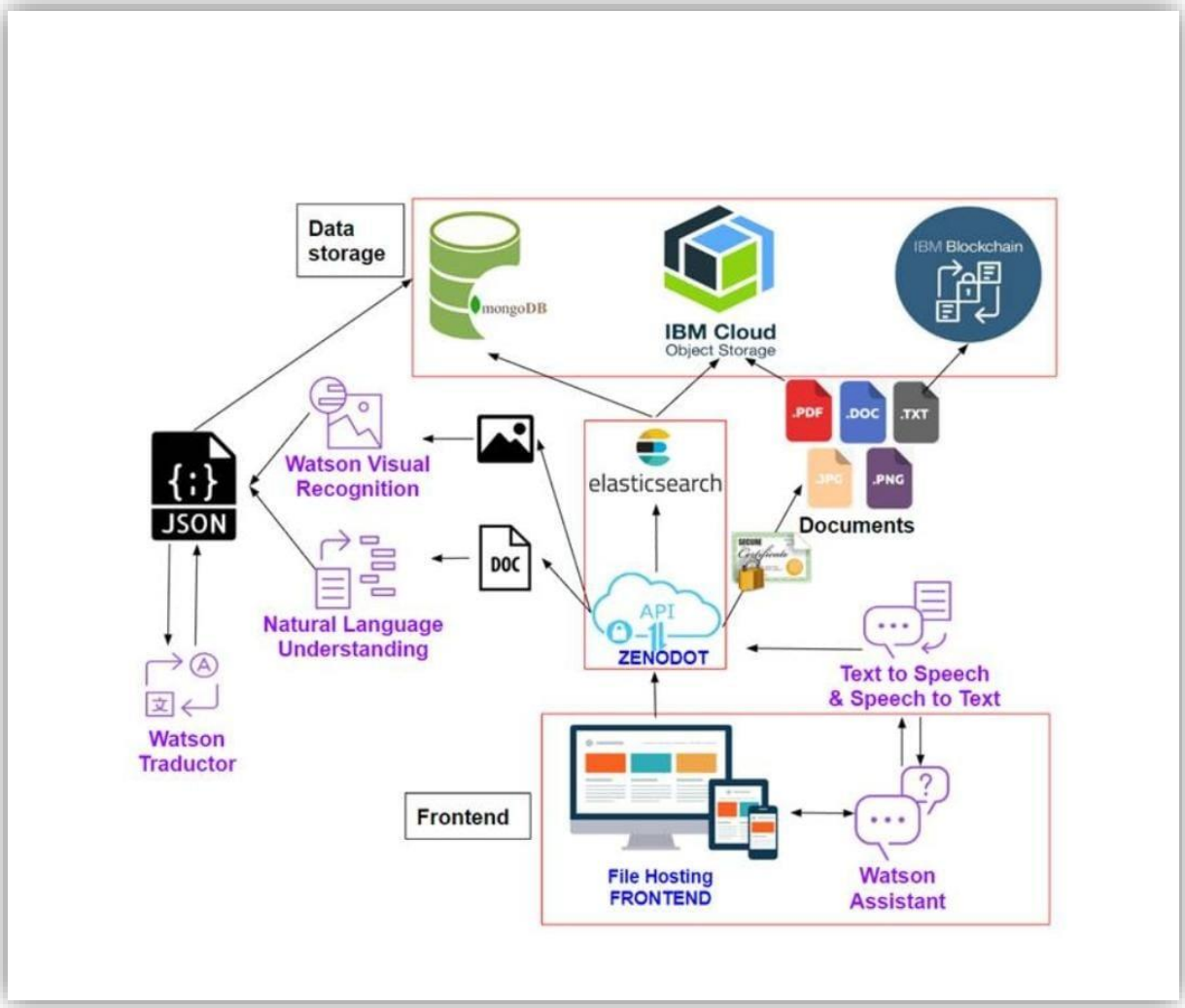
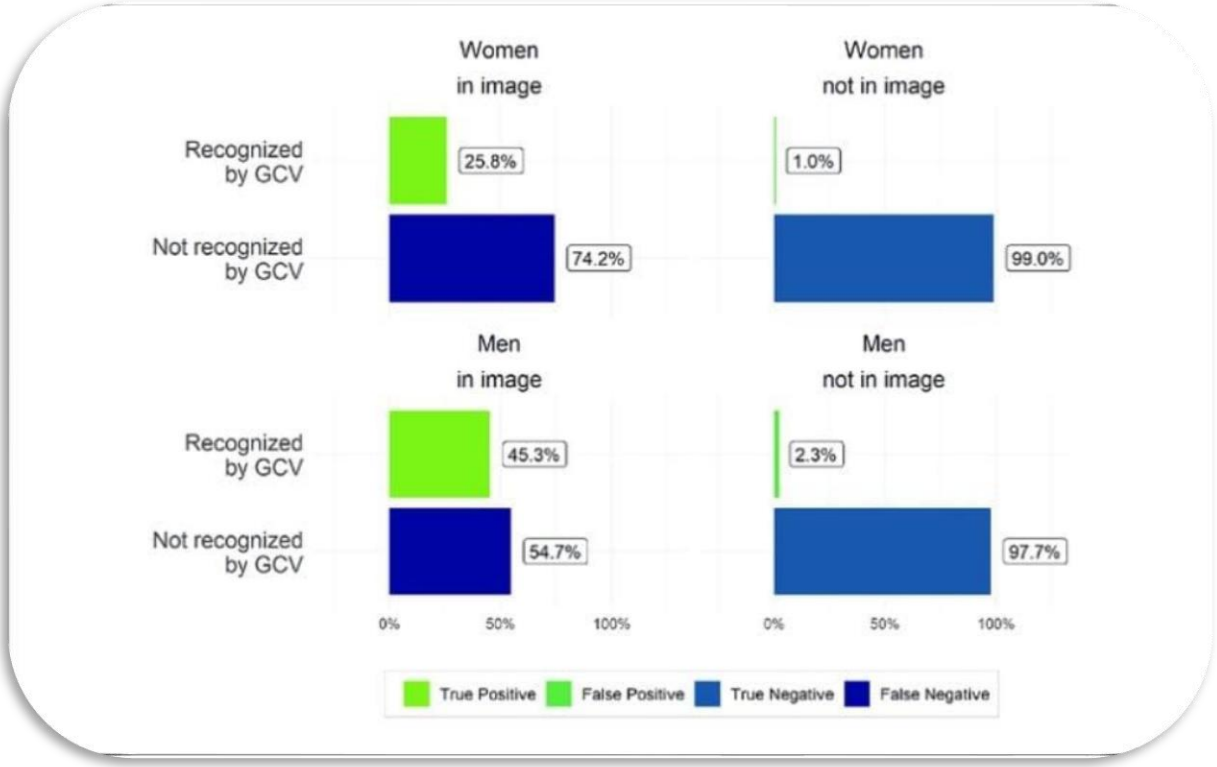


fig: Role of Watson visual recognition services

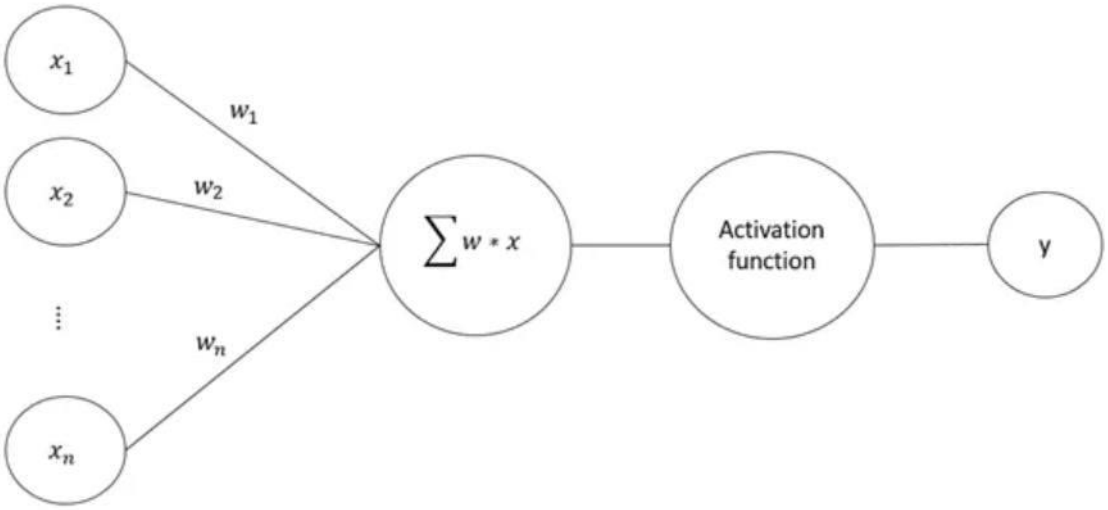
INNOVATION AND FEATURES:

Innovation in image recognition with cloud visual recognition services has been advancing rapidly in recent years. Cloud providers like Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and IBM Watson offer robust tools and services for image. Here are some key areas of innovation and feature development in this field:

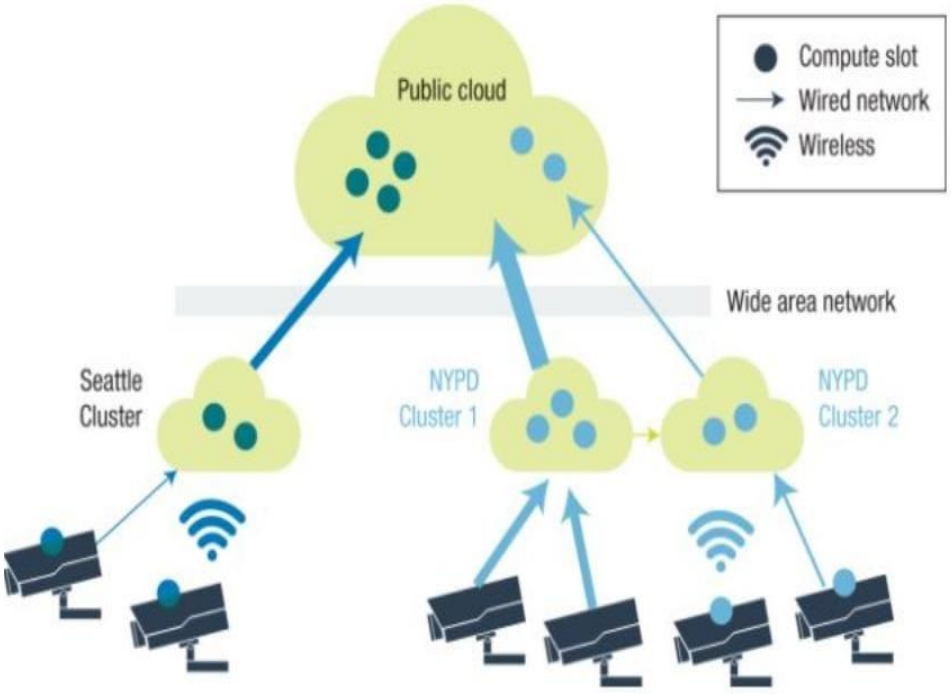
1. **Improved Accuracy**: Continuous research and development efforts have led to significant improvements in image recognition accuracy. Machine learning models are becoming more sophisticated, allowing for better recognition of objects, scenes, and even fine-grained details within images.



2. **Advanced Deep Learning Models**: Cloud visual recognition services have incorporated state-of-the-art deep learning models, such as convolutional neural networks (CNNs) and transformer-based models, to improve image recognition accuracy and handle more complex visual data.



3.Real-time Video Analysis: Cloud visual recognition services have improved their capabilities for real-time video analysis, enabling applications like video surveillance, live event analysis, and sports analytics.



MODEL ARCHITECTURE FOR IMAGE PROCESSING AND VISUAL RECOGNITION:

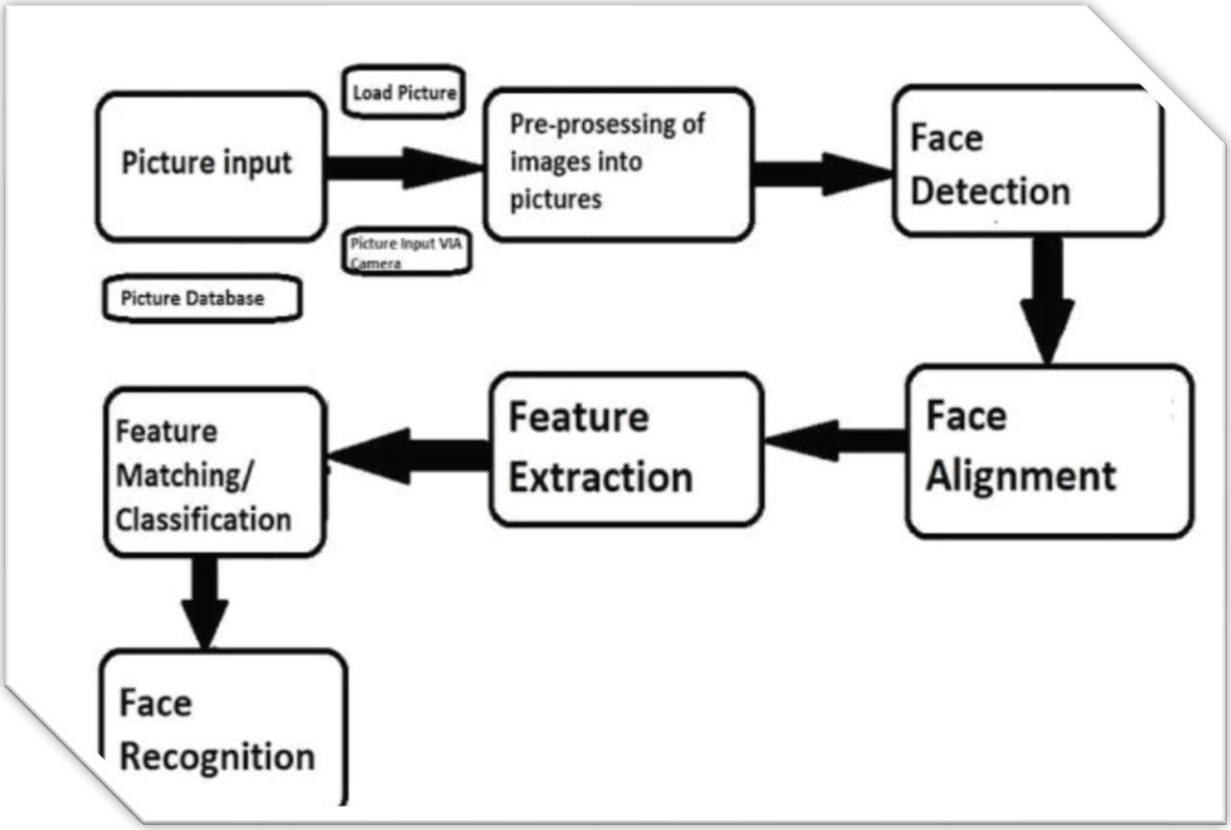


Fig: image processing

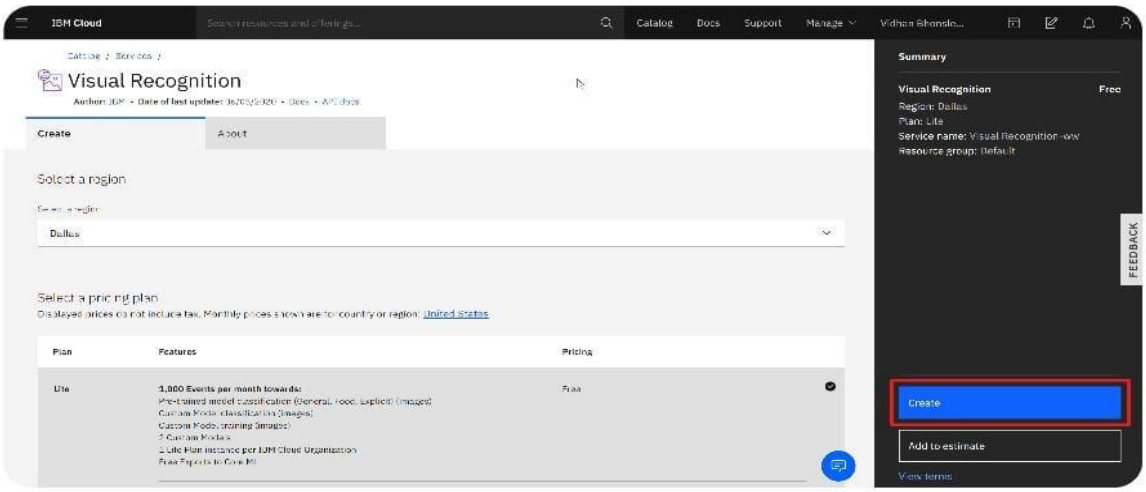


Fig: visual recognition

LINK: https://www.here.com/learn/blog/visual-recognition-with-ibm-watson-here-and-python-part-1

Steps to perform Image Recognition with IBM Cloud Visual Recognition:

1) Set IBM Cloud Account

2)Create a Visual Recognition Service

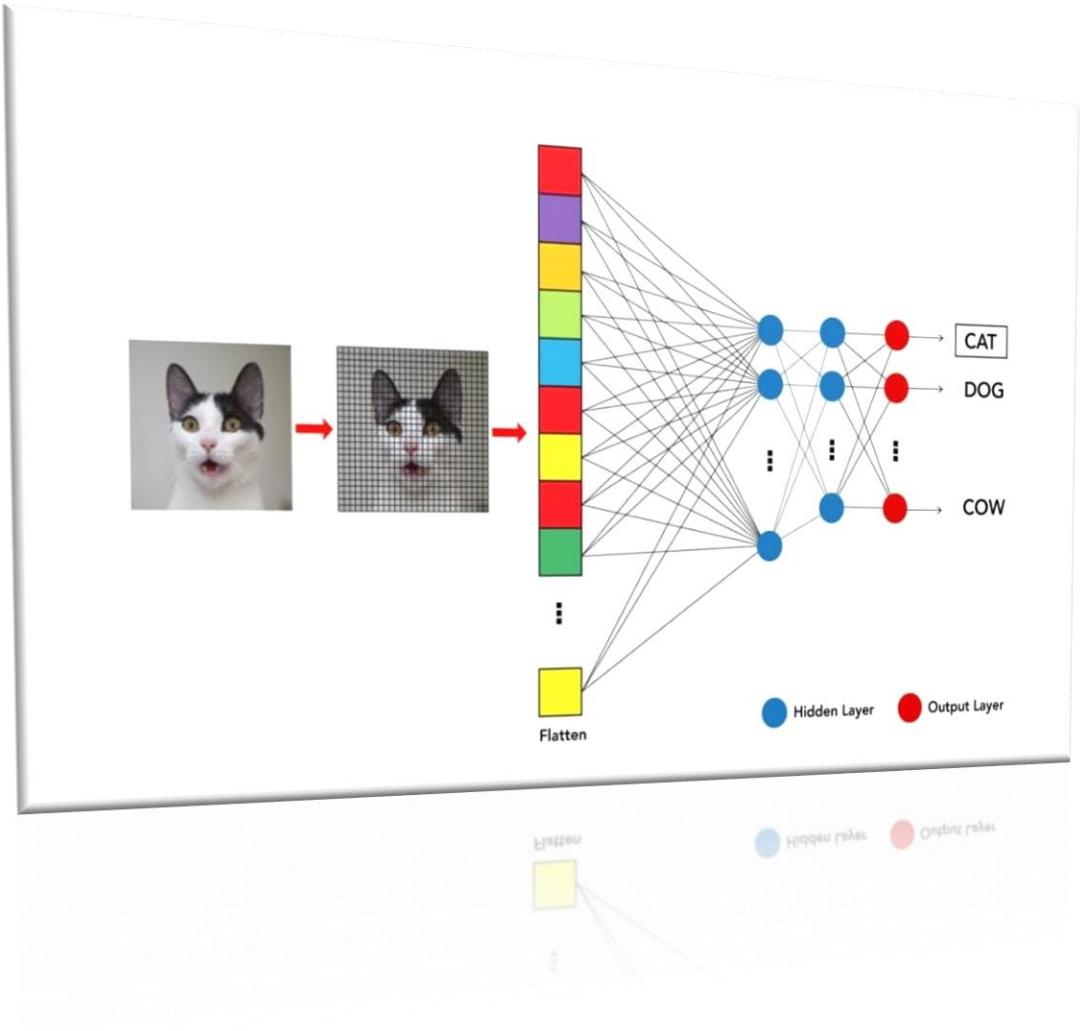
3) Get Your API Key

4) Train a Custom Model(optional)

5) Use the Visual Recognition API

6) Interpret the results

7) Integrate into your Application



For better understanding let see some code of python for image recognition (* NOTE: To perform image recognition with IBM Cloud visual recognition you will need to use the IBM Watson Visual recognition service.)

```
from google.colab import files # Install Kaggle library !pip
install -q kaggle from google.colab import files #upload the
kaggle.json file uploaded = files.upload() #make a diectoryin
which kajggle.json is stored # ! mkdir ~/.kaggle ! cp kaggle.json
~/.kaggle/ #download the dataset into the colab !kaggle datasets
download -d alessiocorrado99/animals10 #unzip the data !unzip
/content/animals10.zip import tensorflow as tf from
tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Input, Dense from
tensorflow.keras import Sequential,Model from
tensorflow.keras.layers import BatchNormalization,Dropout,Flatten
from tensorflow.keras.layers import Conv2D from
tensorflow.keras.layers import MaxPooling2D from
tensorflow.keras.layers import GlobalAveragePooling2D from
tensorflow.keras.preprocessing import image from tensorflow.keras
.layers import GlobalAveragePooling2D import numpy as np import
os import cv2 train_data_dir='/kaggle/input/animals10/raw-img/'
img_height=299 img_width=299 batch_size=64 nb_epochs=20
train_datagen = ImageDataGenerator(rescale=1./255,
shear_range=0.2, zoom_range=0.2, horizontal_flip=True,
validation_split=0.2) # set validation split train_generator =
train_datagen.flow_from_directory( train_data_dir,
target_size=(img_height, img_width), batch_size=batch_size,
class_mode='categorical', subset='training') # set as training
data validation_generator = train_datagen.flow_from_directory(
train_data_dir, # same directory as training data
target_size=(img_height, img_width), batch_size=batch_size,
class_mode='categorical', subset='validation') # set as
validation data #import a pre-trained model, without the top
layers.We will customise #the top layers for our problem
base_model = tf.keras.applications.Xception(include_top=False,
input_shape=(299,299,3)) #For now freeze the initial layers and
do not train them for layer in base_model.layers: layer.trainable
= False # create a custom top classifier x = base_model.output x
= GlobalAveragePooling2D()(x) x = Dense(516,
activation='relu')(x) #since our problem has 10 differnt animals
we have 10 classes #thus we keep 10 nodes in the last layer
predictions = Dense(10, activation='softmax')(x) model =
Model(inputs=base_model.inputs, outputs=predictions)
```

```
model.summary() model.compile(optimizer="adam",
loss="categorical_crossentropy", metrics=["accuracy"])
model.fit_generator( train_generator, steps_per_epoch =
train_generator.samples // batch_size, validation_data =
validation_generator, validation_steps =
validation_generator.samples // batch_size, epochs = nb_epochs)
#Now unfreeze the layers and train the whole model for layer in
base_model.layers: layer.trainable = True history
=model.fit_generator( train_generator, steps_per_epoch =
train_generator.samples // batch_size, validation_data =
validation_generator, validation_steps =
validation_generator.samples // batch_size, epochs = nb_epochs)
model.save('path\name of model') #order of the animals array is
important #animals=["dog", "horse","elephant", "butterfly",
"chicken", "cat", "cow", "sheep","spider", "squirrel"]
bio_animals=sorted(os.listdir('/content/raw-img')) categories =
{'cane': 'dog', "cavallo": "horse", "elefante": "elephant",
"farfalla": "butterfly", "gallina": "chicken", "gatto": "cat",
"mucca": "cow", "pecora": "sheep", "scoiattolo":
"squirrel","ragno":"spider"} def recognise(pred):
animals=[categories.get(item,item) for item in bio_animals]
print("The image consist of ",animals[pred]) from
tensorflow.keras.preprocessing import image import numpy as np
img =
image.load_img("https://d1m75rqqgidzqn.cloudfront.net/kaggle/inpu
t/testttt/OIF-e2bexWrojgtQnAPPcUfOWQ.jpeg", target_size=(299,
299)) x = image.img_to_array(img) x = np.expand_dims(x, axis=0)
prediction=model.predict(x) # prediction
recognise(np.argmax(prediction)) test_data_path="/content/test
data/test_animals" files=sorted(os.listdir(test_data_path))
files=files[1:] for img in files:
x=cv2.imread(os.path.join(test_data_path,img)) cv2_imshow(x)
recognise(np.argmax(predict[files.index(img)])) print("")
```

ANALYSIS AND RESULT:

IBM Watson Visual recognition is a powerful tool that can enhance image analysis workflows. This cloud-based services uses deep learning algorithms to identify object, faces and scenes in images. It can also detect text and recognition logos. Thus, the result in the technology as developed as AI.

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. With a little bit of preprocessing, your application can improve the detection accuracy of the Watson Visual Recognition service, especially when applied to smaller details within an image. Are you ready to learn more? Check out these references for more information:

- Source code: Visual-Recognition-Tile-Localization
- Best Practices for Custom Classifiers
- Watson Visual Recognition Service Documentation

References:

1. https://www.ibm.com/cloud
2. https://www.v7labs.com/blog/image-recognition-guide
3. https://cloud.google.com/vision