**IMAGE RECOGNITION WITH IBM CLOUD VISUAL RECOGNITION**

**INTRODUCTION:-**

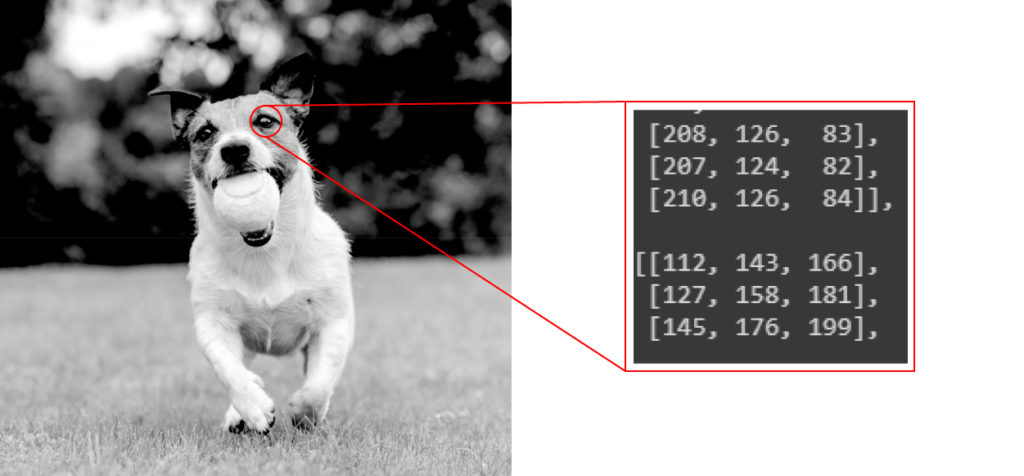
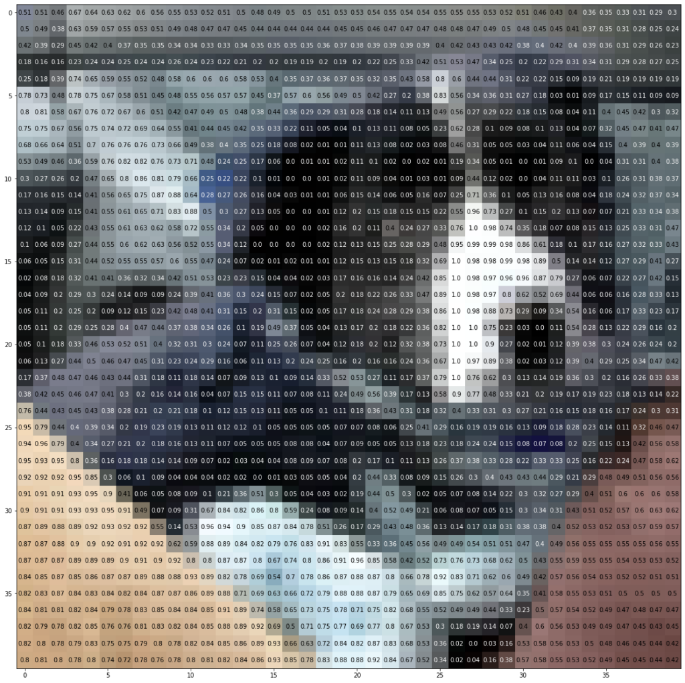


Image recognition refers to technologies that identify places, logos, people, objects, buildings, and several other variables in digital images. It may be very easy for humans like you and me to recognise different images, such as images of animals.

ssWe can easily recognise the image of a cat and differentiate it from an image of a horse. But it may not be so simple for a computer. An image of a dog represented by 40 x 40 pixels.

Image recognition should not be confused with object detection. In object detection, we analyse an image and find different objects in the image while image recognition deals with recognising the images and classifying them into various categories.

**So let me break the process for you in some simple steps:**

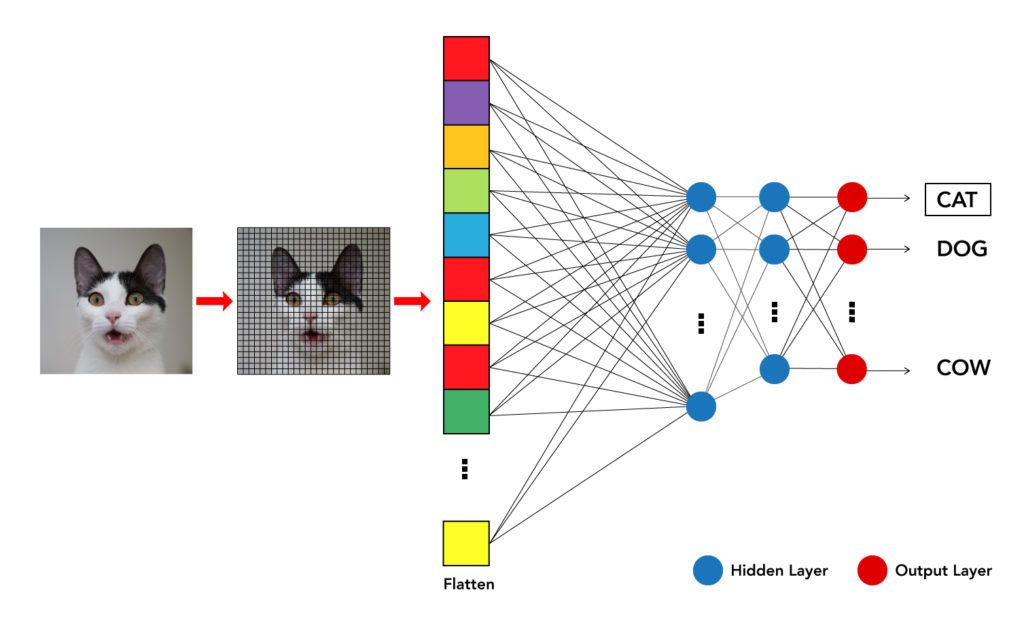


We need a dataset containing images with their respective labels. For example, an image of a dog must be labelled as a dog or something that we can understand.

1. Next, these images are to be fed into a Neural Network and then trained on them. Usually, for the tasks concerned with images, we use [convolutional neural network](https://www.mygreatlearning.com/academy/learn-for-free/courses/convolutional-neural-networks?gl_blog_id=15842). These networks consist of convolutional layers and pooling layers in addition to Multiperceptron layers(MLP). The working of convolutional and pooling layers are explained in the below.
2. We feed in the image that is not in the training set and get predictions.

In the coming sections, by following these simple steps we will make a classifier that can recognise RGB images of 10 different kinds of animals.

**Image Recognition with a pre-trained model**



## **Image Recognition with a pre-trained model**

In this example, I am going to use the Exception model that has been pre-trained on Image net dataset. This technique is basically called [Transfer learning](https://www.mygreatlearning.com/blog/transfer-learning/).

Exception Model is proposed by Francois Chollet. Exception is an extension of the inception Architecture which replaces the standard Inception modules with depth wise Separable Convolutions. This model is available on keras and we just need to import it. So let’s start coding

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/input/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:]

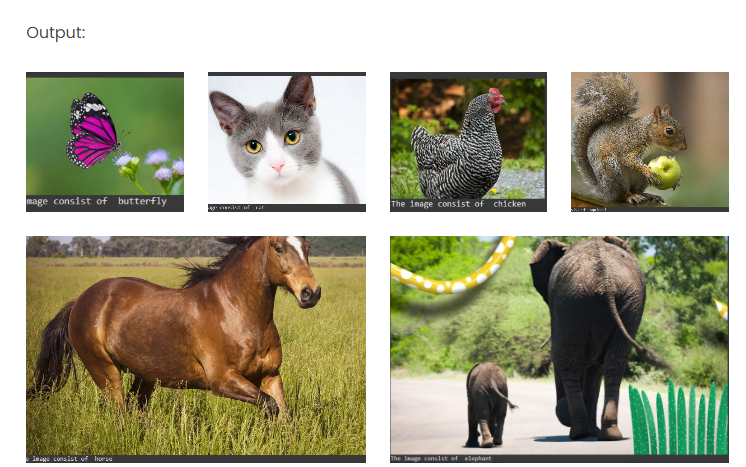
for img in files:

x=cv2.imread(os.path.join(test\_data\_path,img))

cv2\_imshow(x)

recognise(np.argmax(predict[files.index(img)]))

print("")



SUMMARY:-

From this we conclude that IBM Cloud Visual Recognition employs machine learning algorithms to analyze and identify objects and patterns within images. It enables developers to build applications that can understand and interpret visual information, making it particularly valuable in scenarios where automated image analysis is required.