IMAGE RECOGNITION WITH IBM CLOUD VISUAL RECOGNITIONS

DEVELOPMENT PART 2 :

AI & ADS :

Image recognition with deep learning is a key application of AI vision and is used to power a wide range of real-world use cases today. In the following, we will provide a comprehensive overview about the state of the art methods and implementations of image recognition technology.

Image Recognition is the task of identifying objects of interest within an image and recognizing which category the image belongs to. Image recognition, photo recognition, and picture recognition are terms that are used interchangeably. When we visually see an object or scene, we automatically identify objects as different instances and associate them with individual definitions.

However, visual recognition is a highly complex task for machines to perform, requiring significant processing power. Image recognition with artificial intelligence is a long-standing research problem in the computer vision field. While different methods to imitate human vision evolved over time, the common goal of image recognition is the classification of detected objects into different categories (determining the category to which an image belongs).

Therefore, it is also called object recognition. In past years, machine learning, in particular deep learning technology, has achieved big successes in many computer vision and image understanding tasks. Hence, deep learning image recognition methods achieve the best results in terms of performance (computed frames per second/FPS) and flexibility. Later in this article, we will cover the best-performing deep learning algorithms and AI models for image recognition.

In the area of Computer Vision, terms such as Segmentation, Classification, Recognition, and Detection are often used interchangeably, and the different tasks overlap. While this is mostly unproblematic, things get confusing if your workflow requires you to specifically perform a particular task.

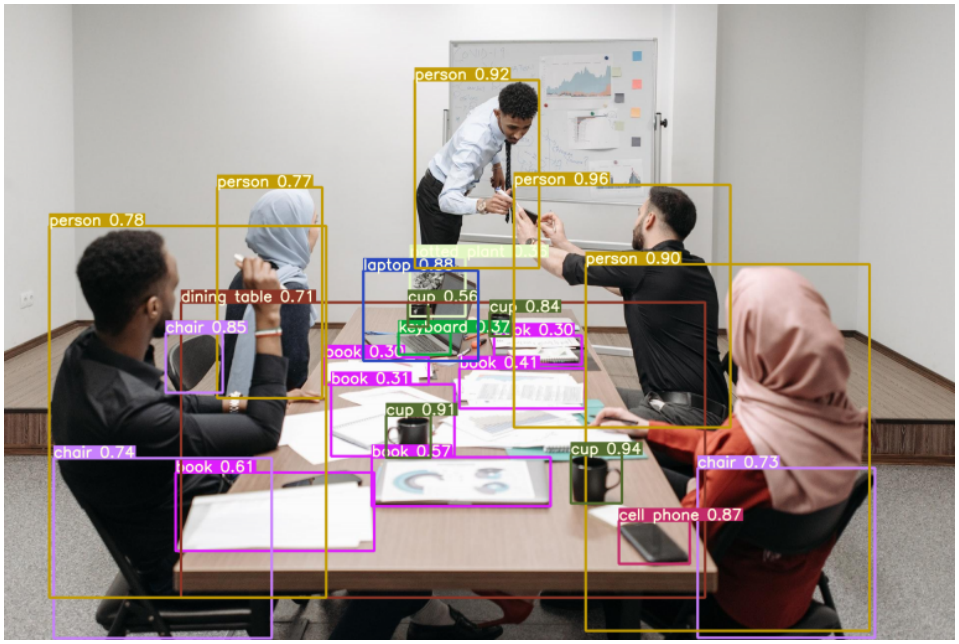


Image Recognition vs. Computer Vision

The terms image recognition and computer vision are often used interchangeably but are actually different. In fact, image recognition is an application of computer vision that often requires more than one computer vision task, such as object detection, image identification, and image classification.

Image Detection is the task of taking an image as input and finding various objects within it. An example is face detection, where algorithms aim to find face patterns in images (see the example below). When we strictly deal with detection, we do not care whether the detected objects are significant in any way.

The goal of image detection is only to distinguish one object from another to determine how many distinct entities are present within the picture. Thus, bounding boxes are drawn around each separate object.

Compared to the traditional computer vision approach in early image processing 20 years ago, deep learning requires only engineering knowledge of a machine learning tool, not expertise in specific machine vision areas to create handcrafted features. While early methods required enormous amounts of training data, newer deep learning methods only need tens of learning samples. However, deep learning requires manual labeling of data to

annotate good and bad samples, a process called image annotation. The process of learning from data that is labeled by humans is called supervised learn.

The Process of Image Recognition Systems

There are a few steps that are at the backbone of how image recognition systems work. Dataset with training data The image recognition models require training data (video, picture, photo, etc.). Neural networks need those training images from an acquired dataset to create perceptions of how certain classes look. For example, an image recognition model that detects different poses (pose estimation model) would need multiple instances of different human poses to understand what makes poses unique from each other.

Training of Neural Networks for Image Recognition The images from the created dataset are fed into a neural network algorithm. This is the deep or machine learning aspect of creating an image recognition model. The training of an image recognition algorithm makes it possible for convolutional neural networks image recognition to identify specific classes. There are multiple well-tested frameworks that are widely used for these purposes today.

AI Model Testing The trained model needs to be tested with images that are not part of the training dataset. This is used to determine the usability, performance, and accuracy of the model. Therefore, about 80-90% of the complete image dataset is used for model training, while the remaining data is reserved for model testing.

DAC :

The DAC consists of multi-scale cavity convolution, and DAC cascades cavity convolution with different expansion rates to extract feature maps with different perceptual fields, adaptively combining local features with their global dependencies to facilitate the capture and characterization of fine feature information. The CLGD fuses cross-level gating mechanisms to guide for selectively enhancing spatial detail and suppressing other regions. Experimental validation on HRF and Cityscapes datasets is conducted, and the experimental results show the effectiveness and robustness of the model in this paper on the defaced image segmentation dataset.

The digital stream is then divided into network packets where it may be sent along with other digital data, not necessarily audio. The packets are then received at the destination, but each packet may take a completely different route and may not even arrive at the destination in the correct time order.

Discrete DACs (circuits constructed from multiple discrete electronic components instead of a packaged IC) would typically be extremely high-speed low-resolution power-hungry types, as used in military radar systems. Very high-speed test equipment, especially sampling oscilloscopes, may also use discrete DACs.

IOT :

In an IoT context, security challenges, including privacy, secure storage, authorization, communication, access control, and administration, are fundamental and complex problems.

The widespread stabilization of IoT devices and services leaves the nodes vulnerable to attacks. Standard security models suffer from several drawbacks and frequently fail to identify the physical dangers in the network because of the restricted computing power.

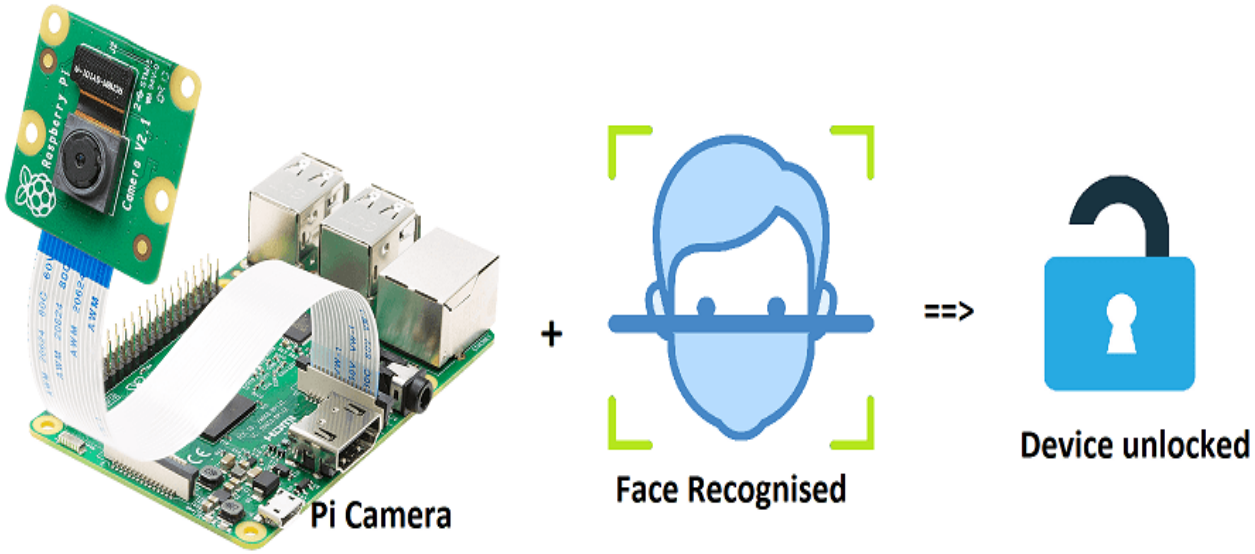
Since only the authorized user should have access to the data and update it often for intelligent applications, IoT security should be improved with ensured connectivity. The finest algorithms for IoT Security resources are cryptography algorithms. In this context, traditional encryption algorithms provide data processing and security.

These algorithms need large mathematical operations and need large memory and power. They are, therefore, not suitable for encryption on IoT devices. However, these devices are currently widely utilized in the Internet of Things, which balances performance and security using lightweight encryption.

The IoT Security device is becoming more significant with the growing number of users and services in IoT networks. The effectiveness of smart objects is increased by the fusion of IoT devices and intelligent environments. However, in crucial smart environments employed in sectors like intelligent irrigation and industrial, the effects of IoT security flaws are quite catastrophic.

Services and Applications in IoT-based intelligent environments will be threatened if security weaker information security in IoT applications necessitates additional study to address these problems since availability, integrity, and secrecy are three fundamental security principles of services and applications based IoT- in intelligent environments.

Finally, a statistical analysis of published publications on security IoT aspects based on cryptographic algorithms and an analysis of the attack type and IoT security challenges was provided. Academics will benefit from this study's help in better comprehending the potential of published research in a particular area of interest.



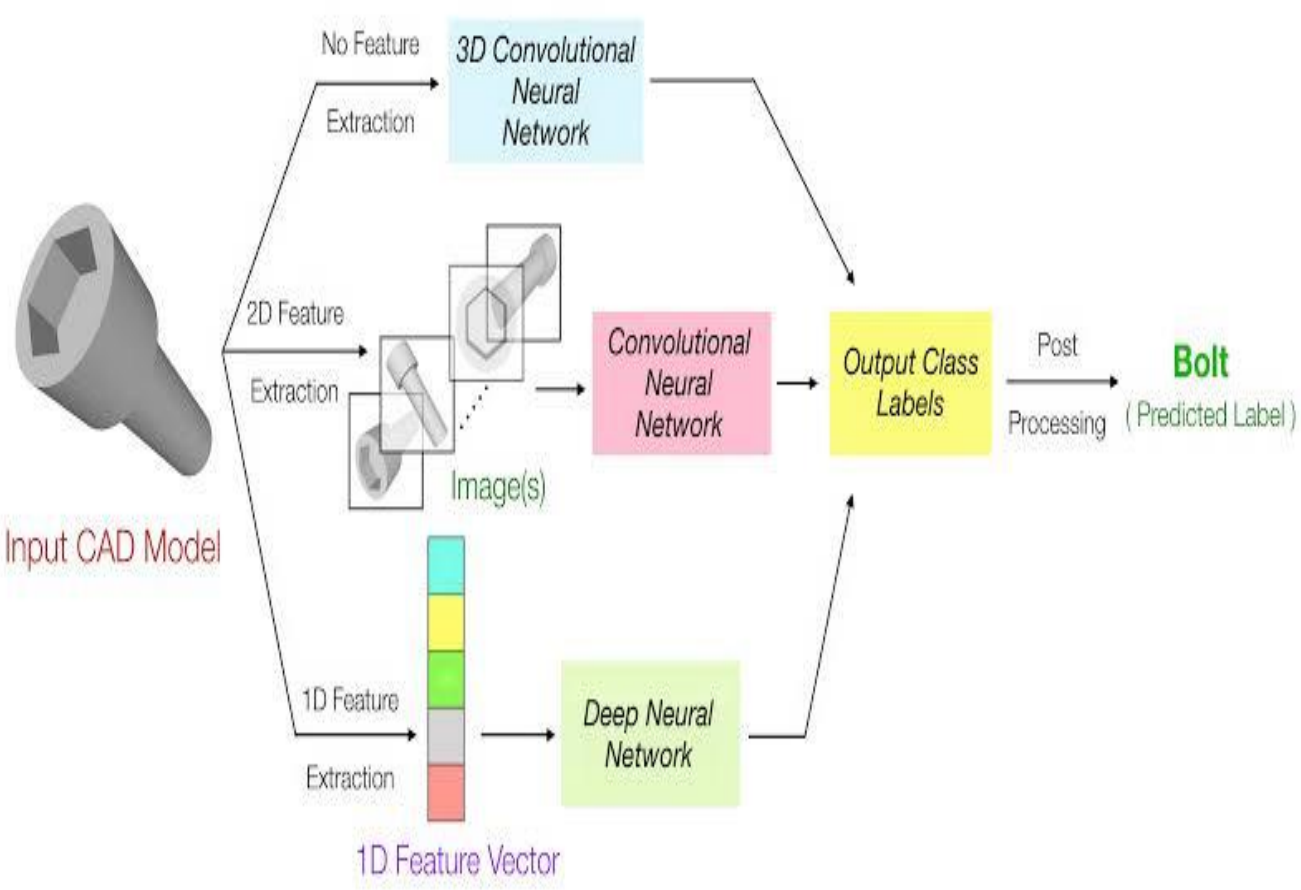
CAD :

The radiologists find it difficult to interpret the digital mammography; hence, computer-aided diagnosis (CAD) technology helps to improve the performance of radiologists by increasing sensitivity rate in a cost-effective way. Current research is focused toward the designing and development of medical imaging and analysis system by using digital image processing tools and the techniques of artificial intelligence, which can detect the abnormality features, classify them, and provide visual proofs to the radiologists.

The computer-based techniques are more suitable for detection of mass in mammography, feature extraction, and classification. The proposed CAD system addresses the several steps such as preprocessing, segmentation, feature extraction, and classification. Though commercial CAD systems are available, identification of subtle signs for breast cancer detection and classification remains difficult. The proposed system presents some advanced techniques in medical imaging to overcome these difficulties.

CAD usually restricted to marking the visible parts or structures in image, whereas CADx helps to evaluate the structures identified in CADe. Both together the CAD models are more significant in identifying the abnormality at an earliest. For example, it highlights

microcalcification clusters, marginal structure of mass, and highly dense structure of tissue in mammography. This helps the radiologist to draw the conclusion. Though the CAD has been used for over 40 years, still it does not reach the expected outcomes. We agree that CAD cannot substitute the doctor but definitely it makes radiologists as better decision makers. It plays a supporting and final interpretative role in medical diagnosis.



PROGRAM :

```
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"])

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 =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(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("")
```

OUTPUT :

THANK YOU