**AI Optical Character Recognition and Business Benefits**

**Optical Character Recognition**

OCR stands for Optical Character Recognition. It is used to recognize text inside images, such as scanned documents and photos. OCR technology is used to convert virtually any kind of image containing written text into machine-readable text data. In addition, the text can be easily translated into multiple languages, making it easy for anyone to understand. However, OCR is not limited to extracting text from document images.

**Why OCR is important**

Even though OCR is not yet 100% accurate, its use cases are growing with the development of deep learning and computer vision. Today, one or another type of OCR is used in retail, communications, finance, healthcare, security, tourism, and other industries.  It is often used as a hidden or silent technology, powering many well-known systems and services in our daily life. It’s used in data entry automation, indexing documents for search engines, automatic number plate recognition, as well as assisting blind and visually impaired people.

**Benefits**

**Number Plate Recognition with OCR**

Automatic Number Plate Recognition (ANPR) uses OCR technology to identify numbers on license plates. These kinds of number-plate identifications are used in a variety of business applications for locating stolen cars, calculating parking fees, controlling access to invoice tolls or security areas, and more.

**OCR Applications in Banking**

The banking industry is deemed one of the largest consumers of OCR technology as it helps enhance security, improves data management, optimizes risk management, and enhances customer experience. Prior to the implementation of OCR technology, most banking documents, including customer records, checks, bank statements and more, were physical. With OCR technology, it has become possible to digitize old documents and store them in a database. OCR technology has completely revolutionized the banking industry by providing easy certification. OCR allows real-time verification of money deposited checks and signing them by scanning them using the OCR based application. An example of this can be seen in the mobile banking app, where checks can be deposited digitally and processed within days through OCR-based check deposit facilities. Enhancing security as electronic deposition of checks through OCR technology results in fraud prevention and increasingly secure transactions, fostering a better user experience.

**Challenges in OCR**

OCR result is very dependent on the quality of the original image, which is why the image pre-processing stage is so important. Common OCR errors include misspellings of letters, lack of readable letters, or the addition of text from adjacent columns. Commonly used methods to normalize an image are to align and rotate the document, remove blur and applying filters, and deleting elements that are not characters (like tables, separator lines, etc.). Elements such as small dots or sharp edges that make up the background can often be read as characters and distort the results of the text recognition process. That’s why the pre-processing stage for OCR should include noise removal and text field isolation. These days OCR systems use computer vision-based algorithms trained on augmented data sets to overcome the presence of noise in the background such as dots, scratches, stains, etc. Augmented datasets are simply datasets that have artificially added sound to teach the OCR model how to deal with noise properly.

**Example**

Example covers OCR recognition using mnist data set.

CELL1:

import tensorflow as tf

from matplotlib import pyplot as plt

import numpy as np

CELL2:

from keras.datasets import mnist

objects=mnist

(train\_img,train\_lab),(test\_img,test\_lab)=objects.load\_data()

CELL3:

for i in range(20):

  plt.subplot(4,5,i+1)

  plt.imshow(train\_img[i],cmap='gray\_r')

  plt.title("Digit : {}".format(train\_lab[i]))

  plt.subplots\_adjust(hspace=0.5)

  plt.axis('off')

CELL4:

print('Training images shape : ',train\_img.shape)

print('Testing images shape : ',test\_img.shape)

CELL5:

print('How image looks like : ')

print(train\_img[0])

CELL6:

plt.hist(train\_img[0].reshape(784),facecolor='orange')

plt.title('Pixel vs its intensity',fontsize=16)

plt.ylabel('PIXEL')

plt.xlabel('Intensity')

CELL7:

train\_img=train\_img/255.0

test\_img=test\_img/255.0

CELL8:

print('How image looks like after normalising: ')

print(train\_img[0])

CELL9:

plt.hist(train\_img[0].reshape(784),facecolor='orange')

plt.title('Pixel vs its intensity',fontsize=16)

plt.ylabel('PIXEL')

plt.xlabel('Intensity')

CELL10:

from keras.models import Sequential

from keras.layers import Flatten,Dense

model=Sequential()

input\_layer= Flatten(input\_shape=(28,28))

model.add(input\_layer)

hidden\_layer1=Dense(512,activation='relu')

model.add(hidden\_layer1)

hidden\_layer2=Dense(512,activation='relu')

model.add(hidden\_layer2)

output\_layer=Dense(10,activation='softmax')

model.add(output\_layer)

CELL11:

#compiling the sequential model

model.compile(optimizer = 'adam',

              loss = 'sparse\_categorical\_crossentropy',

              metrics=['accuracy'])

CELL12:

model.fit(train\_img,train\_lab,epochs=100)

CELL13:

model.save('project.h5')

CELL14:

loss\_and\_acc=model.evaluate(test\_img,test\_lab,verbose=2)

print("Test Loss", loss\_and\_acc[0])

print("Test Accuracy", loss\_and\_acc[1])

CELL15:

plt.imshow(test\_img[0],cmap='gray\_r')

plt.title('Actual Value: {}'.format(test\_lab[0]))

prediction=model.predict(test\_img)

plt.axis('off')

print('Predicted Value: ',np.argmax(prediction[0]))

if(test\_lab[0]==(np.argmax(prediction[0]))):

  print('Successful prediction')

else:

  print('Unsuccessful prediction')

CELL16:

plt.imshow(test\_img[1],cmap='gray\_r')

plt.title('Actual Value: {}'.format(test\_lab[1]))

prediction=model.predict(test\_img)

plt.axis('off')

print('Predicted Value: ',np.argmax(prediction[1]))

if(test\_lab[1]==(np.argmax(prediction[1]))):

  print('Successful prediction')

else:

  print('Unsuccessful prediction')

CELL17:

plt.imshow(test\_img[2],cmap='gray\_r')

plt.title('Actual Value: {}'.format(test\_lab[2]))

prediction=model.predict(test\_img)

plt.axis('off')

print('Predicted Value: ',np.argmax(prediction[2]))

if(test\_lab[2]==(np.argmax(prediction[2]))):

  print('Successful prediction')

else:

  print('Unsuccessful prediction')