**A Weekly Report**

**on**

**INVOICE MANAGEMENT**

**at**

# **OneOnic Solution**

**by**

**Mr. Tank Viraj Rupeshbhai**

**EC034, 21ECUBG069**

**B.Tech. (EC) Semester - VIII**

**Faculty Supervisor External Guide**

**Dr. NarendraKumar Chauhan Mr. Umesh Ghediya**

**In Partial Fulfillment of Requirement of**

**Bachelor of Technology** - **Electronics & Communication**

**Submitted To**



**Department of Electronics & Communication Engineering**

**Faculty of Technology**

**Dharmsinh Desai University, Nadiad - 387001.**

**(Jan 2025)**

Task 5 :- Comparative Analysis of OCR Tools

Introduction :-

Here, we are learning how computers can read words from pictures using special tools called OCR (Optical Character Recognition). These tools, like Tesseract OCR, PyOCR, and PaddleOCR, help the computer find and understand words in images or scanned papers. Each tool works differently, and we want to see which one is best for different types of text. The goal is to check how fast these tools work, how many words they get right, and how easy they are to use. This will help us choose the best tool for future tasks.

Tool’s and Technology Used :-

1. Tesseract OCR:

A popular, free OCR tool that helps computers read text from images. It is easy to use and works well with clear, printed text.

1. PyOCR :

A lightweight Python library that provides a simple way to use OCR engines. It allows text extraction from images through its easy-to-use interface.

1. PaddleOCR:

A deep-learning-based OCR tool that supports advanced features like multi-language recognition and handling slanted or rotated text.

1. OpenCV:

A library used to process images, such as reading and preparing them for OCR tools.

1. SequenceMatcher:

A Python library used to calculate the accuracy of the extracted text by comparing it with the original text.

1. Matplotlib:

A visualization library used to display images and their OCR results for better comparison and analysis.

Workflow :-

* Prepare the image:

Load the image into the computer and change its colors to make it easier for the tools to read.

* Show the original image:

Display the original image so you can see what the computer is trying to read.

* Use OCR tools to read the image:

First, use Tesseract OCR to read the text and check how long it takes. Then, try PyOCR and measure the time. Finally, use PaddleOCR and note how long it takes.

* Show the text found by each OCR tool:

For each tool, show the text it found and the time taken to extract it.

* Check the accuracy of the text:

Compare the text each tool found with the correct text to see how accurate each tool is.

* Compare the tools:

Look at the results and decide which tool was the most accurate and which one was the fastest.

Code Implementation :-

!pip install pyocr -q

!sudo apt install tesseract-ocr -q

!pip install pytesseract -q

!pip install easyocr -q

!pip install paddlepaddle -q

!pip install paddleocr -q

# Import libraries

import PIL.Image

import pyocr

import pyocr.builders

from paddleocr import PaddleOCR

import pytesseract

import cv2

import time

from difflib import SequenceMatcher

from matplotlib import pyplot as plt

# Function to calculate accuracy

def calculate\_accuracy(reference, extracted):

"""

Calculates accuracy as the similarity ratio between reference and extracted text.

"""

return SequenceMatcher(None, reference, extracted).ratio()

# Tesseract OCR

def extract\_text\_with\_tesseract(image\_path):

"""

Extracts text from an image using Tesseract OCR.

"""

try:

image = cv2.imread(image\_path)

start\_time = time.time()

extracted\_text = pytesseract.image\_to\_string(image)

end\_time = time.time()

time\_taken = end\_time - start\_time

return extracted\_text.strip(), time\_taken

except Exception as e:

return f"Error occurred: {str(e)}", 0

# Initialize the PyOCR engine

tools = pyocr.get\_available\_tools()

if len(tools) == 0:

print("No OCR tool found.")

exit(1)

ocr\_tool = tools[0]

# PyOCR

def extract\_text\_with\_pyocr(image\_path):

"""

Extracts text from an image using PyOCR.

"""

try:

image = PIL.Image.open(image\_path)

start\_time = time.time()

extracted\_text = ocr\_tool.image\_to\_string(image, builder=pyocr.builders.TextBuilder())

end\_time = time.time()

time\_taken = end\_time - start\_time

return extracted\_text.strip(), time\_taken

except Exception as e:

return f"Error occurred: {str(e)}", 0

# Paddle OCR

def extract\_text\_with\_paddleocr(image\_path):

"""

Extracts text from an image using PaddleOCR.

"""

try:

ocr = PaddleOCR(use\_angle\_cls=True, lang='en')

start\_time = time.time()

results = ocr.ocr(image\_path)

end\_time = time.time()

extracted\_text = "\n".join([line[1][0] for line in results[0]])

time\_taken = end\_time - start\_time

return extracted\_text.strip(), time\_taken

except Exception as e:

return f"Error occurred: {str(e)}", 0

# Input Image and Reference Text

image\_path = '/content/drive/MyDrive/Task5/test\_image/ddu\_2.jpg'

reference\_text = "DHARMSINH DESAI UNIVERSITY"

# Read and process the image

img = cv2.imread(image\_path)

# Convert from BGR to RGB

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

print("Original Image")

plt.imshow(img\_rgb)

plt.show()

# Extract text using OCR

print("Using Tesseract OCR...")

tesseract\_text, tesseract\_time = extract\_text\_with\_tesseract(image\_path)

print(f"Tesseract OCR Text:\n{tesseract\_text}")

print(f"Time taken by Tesseract OCR: {tesseract\_time:.2f} seconds")

# Extract text using PyOCR

print("\nUsing PyOCR...")

pyocr\_text, pyocr\_time = extract\_text\_with\_pyocr(image\_path)

print(f"PyOCR Text:\n{pyocr\_text}")

print(f"Time taken by PyOCR: {pyocr\_time:.2f} seconds")

# PaddleOCR

print("\nUsing PaddleOCR...")

paddleocr\_text, paddleocr\_time = extract\_text\_with\_paddleocr(image\_path)

print(f"PaddleOCR Text:\n{paddleocr\_text}")

print(f"Time taken by PaddleOCR: {paddleocr\_time:.2f} seconds")

# Accuracy Comparison

print("\nComparison:")

print(f"Tesseract OCR Accuracy: {calculate\_accuracy(reference\_text, tesseract\_text):.2f}")

print(f"PyOCR Accuracy: {calculate\_accuracy(reference\_text, pyocr\_text):.2f}")

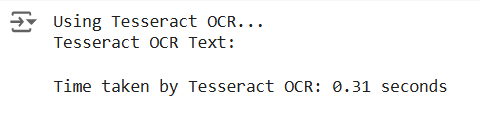
print(f"Paddle OCR Accuracy: {calculate\_accuracy(reference\_text, paddleocr\_text):.2f}")

Results :-

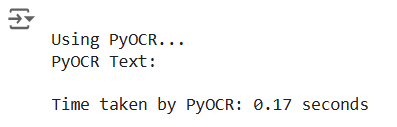
Input Image :-



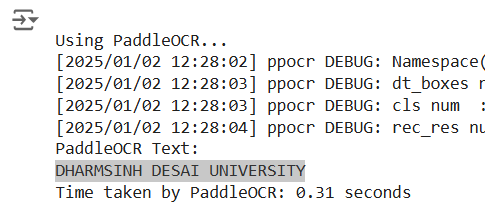
Output Of Tesseract :-



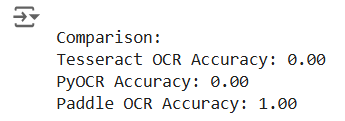
Output Of PyOCR :-



Output of PaddleOCR :-



Final Comparison Of Accuracy of each OCR :-



Observation :-

* Tesseract OCR’s accuracy was 0% and it took 0.31sec.
* PyOCR’s accuracy was also 0% and it took 0.17sec.
* PaddleOCR’s accuracy was 100% and it took 0.31sec.

Conclusion :-

From the tests, we found that Tesseract OCR and PyOCR didn’t extract any text from the image. However, PaddleOCR worked perfectly and got all the text with 100% accuracy. Tesseract and PaddleOCR took the same time of 0.31 seconds, while PyOCR was a little faster at 0.17 seconds.

In this case, PaddleOCR was the best because it read the text correctly and was just as fast as Tesseract. But if the image is a little tougher or has more complicated text, these all tools might not work well. In such cases, we would need to use more advanced OCR tools like KerasOCR or EasyOCR to get better results.

Task 6 :- Advanced OCR Analysis

Introduction :-

In this task, we explore how computers read text from images using OCR tools like Tesseract OCR, EasyOCR, and Keras-OCR. In Task 5, PaddleOCR worked perfectly, while Tesseract and PyOCR failed, showing that some tools work better for specific tasks. Now, we’ll compare these three tools to see which is fastest, most accurate, and easiest to use. The goal is to find the best tool for tasks like digitizing documents or reading text from photos, especially for complex images where advanced tools might be needed. By testing these tools, we aim to understand their strengths and weaknesses, helping us choose the right OCR tool for different real-world applications. This comparison will also give us insights into how well these tools handle challenges like noisy backgrounds or handwritten text.

Tool’s and Technology Used :-

1. Tesseract OCR:

A popular, free OCR tool that helps computers read text from images. It is easy to use and works well with clear, printed text.

1. EasyOCR:

A lightweight and fast OCR tool that supports multiple languages.

1. Keras-OCR:

A deep learning-based OCR tool for more advanced text recognition.

1. OpenCV:

A library used to process images, such as reading and preparing them for OCR tools.

1. Matplotlib:

A visualization library used to display images and their OCR results for better comparison and analysis.

1. Pandas:

A library used to organize and display data in tables.

Workflow :-

* Prepare the image:

Load the image into the computer and change its colors to make it easier for the tools to read.

* Show the original image:

Display the original image so you can see what the computer is trying to read.

* Extract Text Using Tesseract OCR:

Use Tesseract OCR to extract text from both images. Measure the time it takes for each image and save the results.

* Extract Text Using EasyOCR:

Use EasyOCR to extract text and confidence scores from both images. Measure the time it takes and store the results in a table for better organization.

* Extract Text Using KerasOCR:

Use KerasOCR to extract text and bounding boxes from both images. Measure the time and save the results in a table for comparison.

* Display OCR Results:

For each image, show the text extracted by Tesseract, EasyOCR, and KerasOCR. Include the time each tool took to process the image.

* Visualize Results:

Compare the bounding boxes and text visually for EasyOCR and KerasOCR. Use side-by-side plots to see how well each tool detected text in the images.

Code Implementation :-

!pip install keras-ocr --upgrade --force-reinstall -q

!pip install --force-reinstall -v "tensorflow==2.15.1" -q

!pip install easyocr -q

!pip install pytesseract -q

!apt-get install tesseract-ocr -q

# Import libraries

import easyocr

import keras\_ocr

import pytesseract

from pytesseract import Output

import cv2

import time

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

# Tesseract OCR

def extract\_text\_with\_tesseract(image\_path):

"""

Extracts text from an image using Tesseract OCR.

"""

try:

# Read the image

image = cv2.imread(image\_path)

# Perform OCR using Tesseract

start\_time = time.time()

extracted\_text = pytesseract.image\_to\_string(image)

end\_time = time.time()

time\_taken = end\_time - start\_time

return extracted\_text, time\_taken

except Exception as e:

return f"Error occurred: {str(e)}", 0

# Easy OCR

def extract\_with\_easyocr(image\_path):

"""

Extracts text and bounding boxes from an image using EasyOCR and measures time.

"""

reader = easyocr.Reader(['en'], gpu=True)

start\_time = time.time()

results = reader.readtext(image\_path)

end\_time = time.time()

df = pd.DataFrame(results, columns=['bbox', 'text', 'conf'])

time\_taken = end\_time - start\_time

return df, time\_taken

# Keras OCR

def extract\_with\_kerasocr(image\_path):

"""

Extracts text and bounding boxes from an image using KerasOCR and measures time.

"""

pipeline = keras\_ocr.pipeline.Pipeline()

start\_time = time.time()

results = pipeline.recognize([image\_path])

end\_time = time.time()

df = pd.DataFrame(results[0], columns=['text', 'bbox'])

time\_taken = end\_time - start\_time

return df, time\_taken

# Visualization for comparison

def plot\_compare(img\_fn, easyocr\_df, kerasocr\_df):

"""

Plots EasyOCR and KerasOCR results side by side for a single image.

"""

fig, axs = plt.subplots(1, 2, figsize=(15, 10))

# EasyOCR results

easy\_results = easyocr\_df[['text', 'bbox']].values.tolist()

easy\_results = [(x[0], np.array(x[1])) for x in easy\_results]

keras\_ocr.tools.drawAnnotations(plt.imread(img\_fn), easy\_results, ax=axs[0])

axs[0].set\_title('EasyOCR Results', fontsize=24)

# KerasOCR results

keras\_results = kerasocr\_df[['text', 'bbox']].values.tolist()

keras\_results = [(x[0], np.array(x[1])) for x in keras\_results]

keras\_ocr.tools.drawAnnotations(plt.imread(img\_fn), keras\_results, ax=axs[1])

axs[1].set\_title('KerasOCR Results', fontsize=24)

plt.show()

# Path to the input image

image\_path\_1 = "/content/drive/MyDrive/Task5/test\_image/ddu\_2.jpg"

image\_path\_2 = "/content/drive/MyDrive/Task5/test\_image/ddu.jpg"

# Process image

img\_1 = cv2.imread(image\_path\_1)

img\_rgb\_1 = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2RGB)

print("Test Image 1")

plt.imshow(img\_rgb\_1)

plt.show()

img\_2 = cv2.imread(image\_path\_2)

img\_rgb\_2 = cv2.cvtColor(img\_2, cv2.COLOR\_BGR2RGB)

print("\nTest Image 2")

plt.imshow(img\_rgb\_2)

plt.show()

# Tesseract OCR

print("\nUsing Tesseract OCR...")

tesseract\_text, tesseract\_time = extract\_text\_with\_tesseract(image\_path\_1)

print(f"\nTesseract Results For Test Image 1:\n{tesseract\_text}")

print(f"Time taken: {tesseract\_time:.2f} seconds")

tesseract\_text, tesseract\_time = extract\_text\_with\_tesseract(image\_path\_2)

print(f"\nTesseract Results For Test Image 2:\n{tesseract\_text}")

print(f"Time taken: {tesseract\_time:.2f} seconds")

# EasyOCR

print("\nUsing EasyOCR...")

easyocr\_df\_1, easyocr\_time\_1 = extract\_with\_easyocr(image\_path\_1)

print(f"\n\nEasyOCR Results For Test Image 1:\n{easyocr\_df\_1[['text', 'conf']]}")

print(f"Time taken: {easyocr\_time\_1:.2f} seconds")

easyocr\_df\_2, easyocr\_time\_2 = extract\_with\_easyocr(image\_path\_2)

print(f"\n\nEasyOCR Results For Test Image 2:\n{easyocr\_df\_2[['text', 'conf']]}")

print(f"Time taken: {easyocr\_time\_2:.2f} seconds")

# KerasOCR

print("\nUsing KerasOCR...")

kerasocr\_df\_1, kerasocr\_time\_1 = extract\_with\_kerasocr(image\_path\_1)

print(f"\n\nKerasOCR Results For Test Image 1:\n{kerasocr\_df\_1[['text']]}")

print(f"Time taken: {kerasocr\_time\_1:.2f} seconds")

kerasocr\_df\_2, kerasocr\_time\_2 = extract\_with\_kerasocr(image\_path\_2)

print(f"\n\nKerasOCR Results For Test Image 2:\n{kerasocr\_df\_2[['text']]}")

print(f"Time taken: {kerasocr\_time\_2:.2f} seconds")

# Visualization

print("\nVisualizing OCR Results...")

print("\n\n")

plot\_compare(image\_path\_1, easyocr\_df\_1, kerasocr\_df\_1)

print("\n\n")

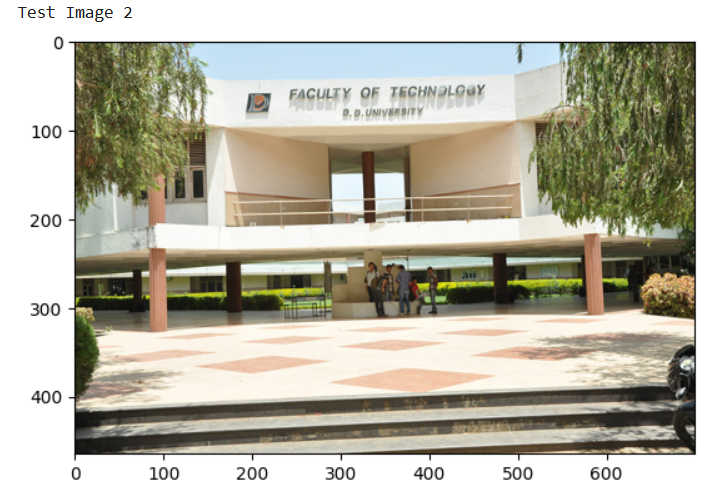
plot\_compare(image\_path\_2, easyocr\_df\_2, kerasocr\_df\_2)

Results :-

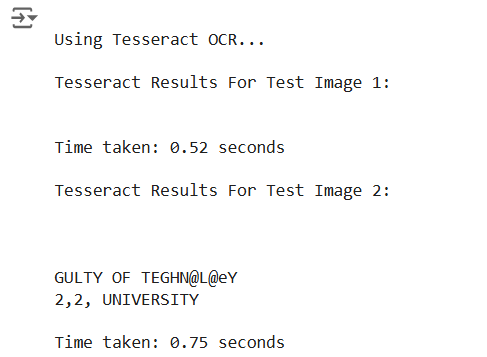
Input Image 1 :-



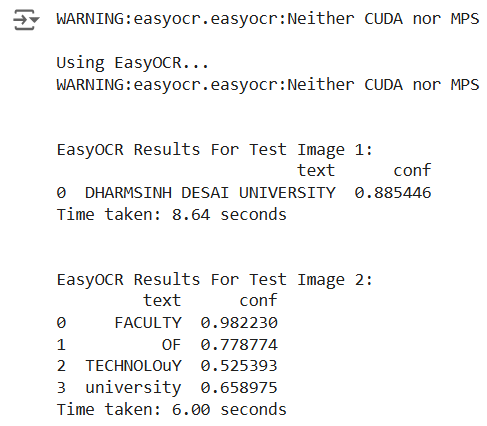
Input Image 2 :-



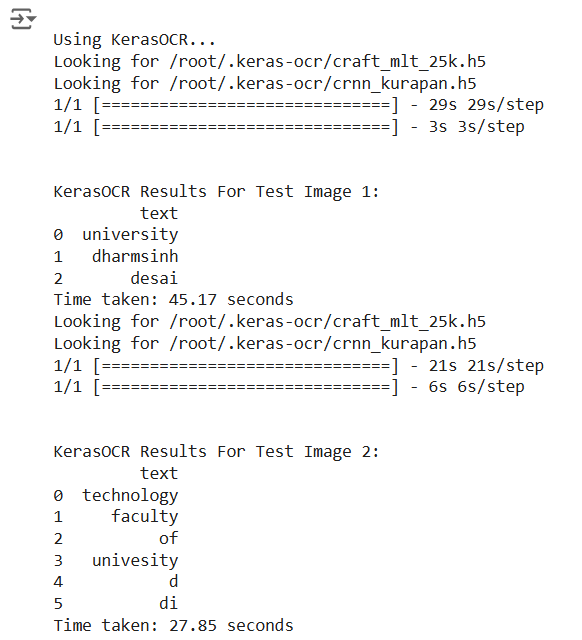
Output Of Tesseract :-



Output Of EasyOCR :-



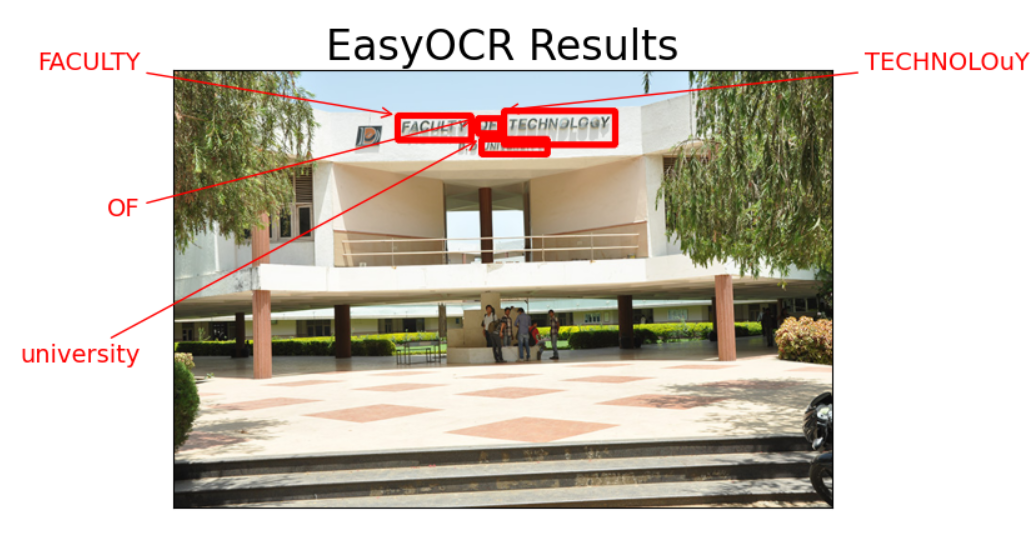
Output Of Keras OCR :-

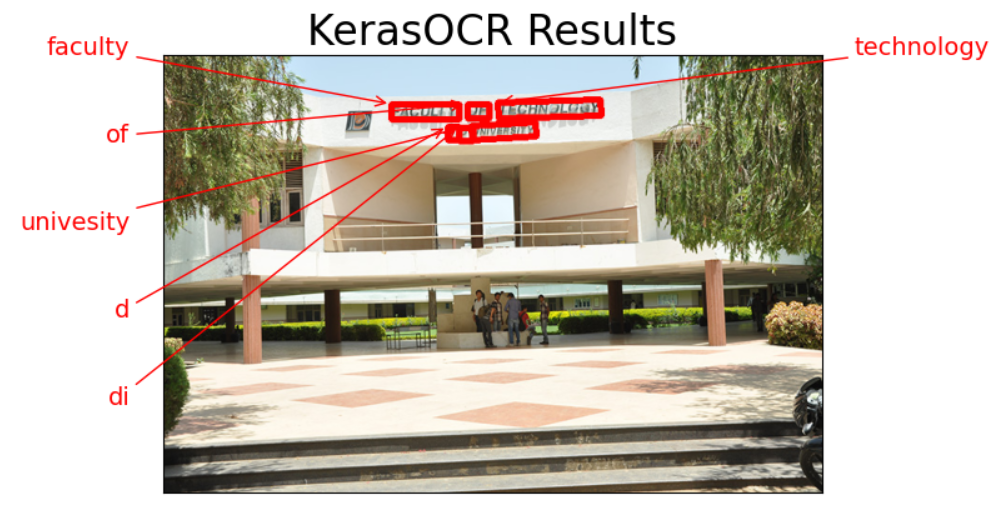


Visualization of how both Engines extract text :-









Observation :-

* Easy OCR took 8.64sec for image 1 and 6sec for image 2.
* Keras OCR took 45.17sec for image 1 and 27.85sec for image 2.
* According to accuracy KerasOCR out performs EasyOCR and According to time taken EasyOCR out performs KerasOCR.

Conclusion :-

From the tests, we found that EasyOCR is much faster, taking 8.64 seconds for Image 1 and 6 seconds for Image 2. In comparison, KerasOCR took longer, with 45.17 seconds for Image 1 and 27.85 seconds for Image 2.

However, when it comes to accuracy, KerasOCR performed better than EasyOCR, making it more reliable for extracting text correctly.

If speed is the priority, EasyOCR is the better choice. But if accuracy is more important, KerasOCR is the preferred tool despite its longer processing time.