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.
2) 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.
3) PaddleOCR:
   A deep-learning-based OCR tool that supports advanced features like multi-language recognition and handling slanted or rotated text.
4) OpenCV:
   A library used to process images, such as reading and preparing them for OCR tools.
5) SequenceMatcher:
   A Python library used to calculate the accuracy of the extracted text by comparing it with the original text.
6) 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.

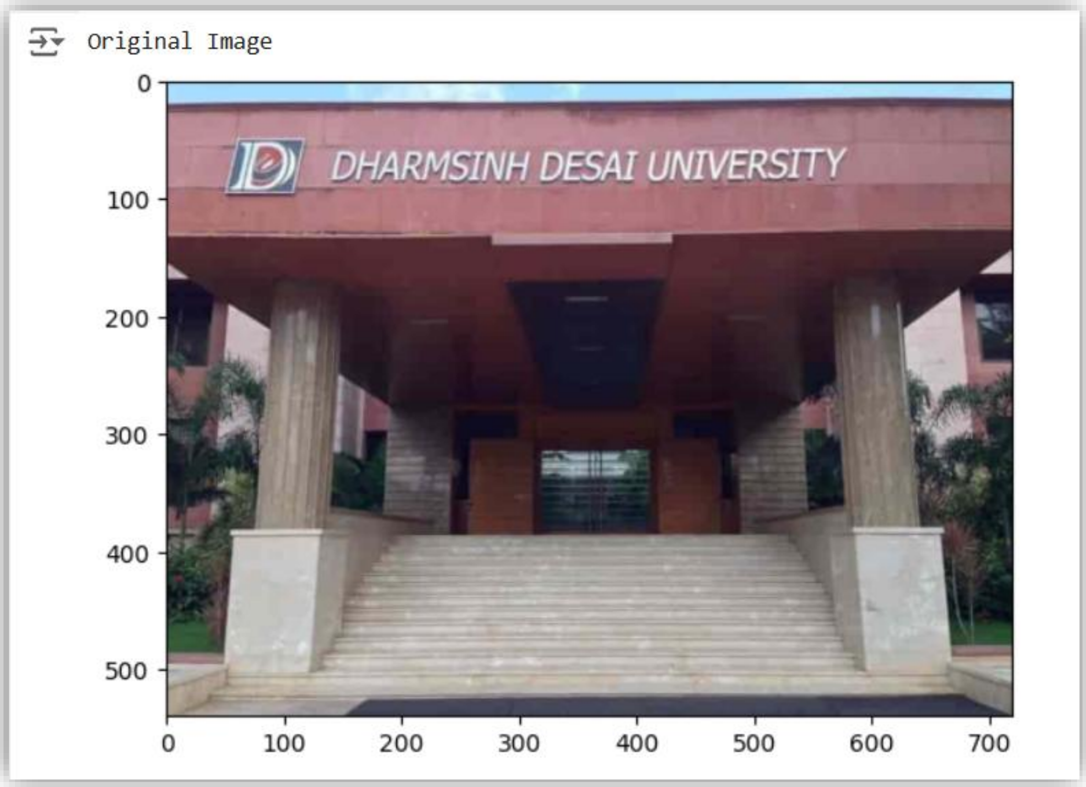
- Compare the tools:
  Look at the results and decide which tool was the most accurate and which one was the fastest.

Code Implementation :-

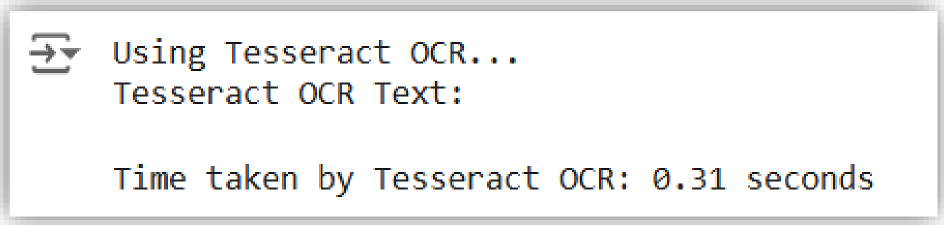
```
!pip install pyocr -q
!sudo apt install tesseract-ocr -q
!pip install pytesseract -q
!pip install easyocr -q
!pip install paddlepaddle -q
!pip install paddleocr -q

# 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 :-

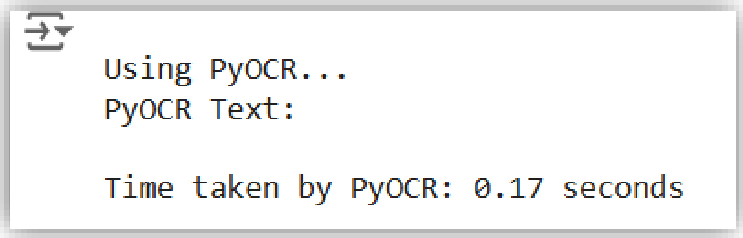


Output Of Tesseract :-



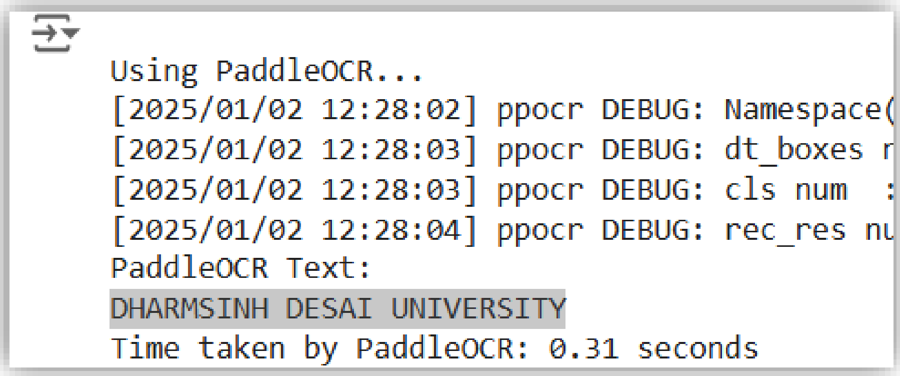
```
Using Tesseract OCR...
Tesseract OCR Text:

Time taken by Tesseract OCR: 0.31 seconds
```

Output Of PyOCR :-



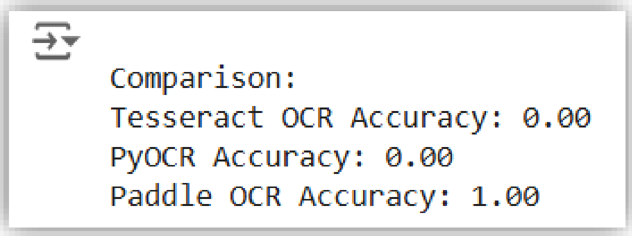
```
Using PyOCR...
PyOCR Text:

Time taken by PyOCR: 0.17 seconds
```

Output of PaddleOCR :-



```
Using PaddleOCR...
[2025/01/02 12:28:02] ppocr DEBUG: Namespace(
[2025/01/02 12:28:03] ppocr DEBUG: dt_boxes r
[2025/01/02 12:28:03] ppocr DEBUG: cls num  :
[2025/01/02 12:28:04] ppocr DEBUG: rec_res nu
PaddleOCR Text:
DHARMSINH DESAI UNIVERSITY
Time taken by PaddleOCR: 0.31 seconds
```

Final Comparison Of Accuracy of each OCR :-



```
Comparison:
Tesseract OCR Accuracy: 0.00
PyOCR Accuracy: 0.00
Paddle OCR Accuracy: 1.00
```

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.

2) EasyOCR:
   A lightweight and fast OCR tool that supports multiple languages.

3) Keras-OCR:
   A deep learning-based OCR tool for more advanced text recognition.

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

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

6) 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 :-

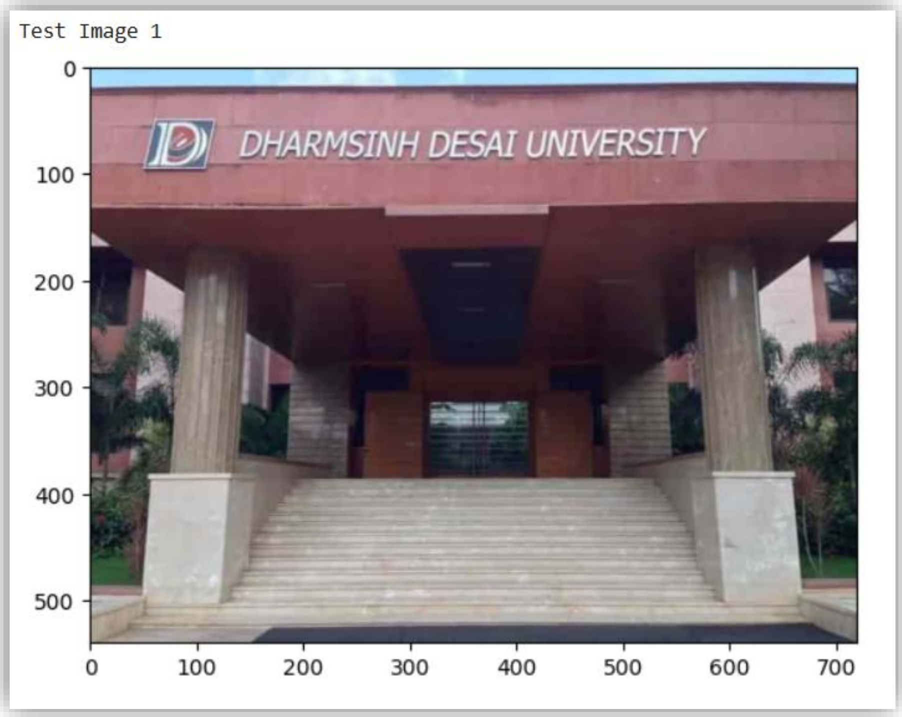
```
!pip install keras-ocr --upgrade --force-reinstall -q
!pip install --force-reinstall -v "tensorflow==2.15.1" -q

!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)
```

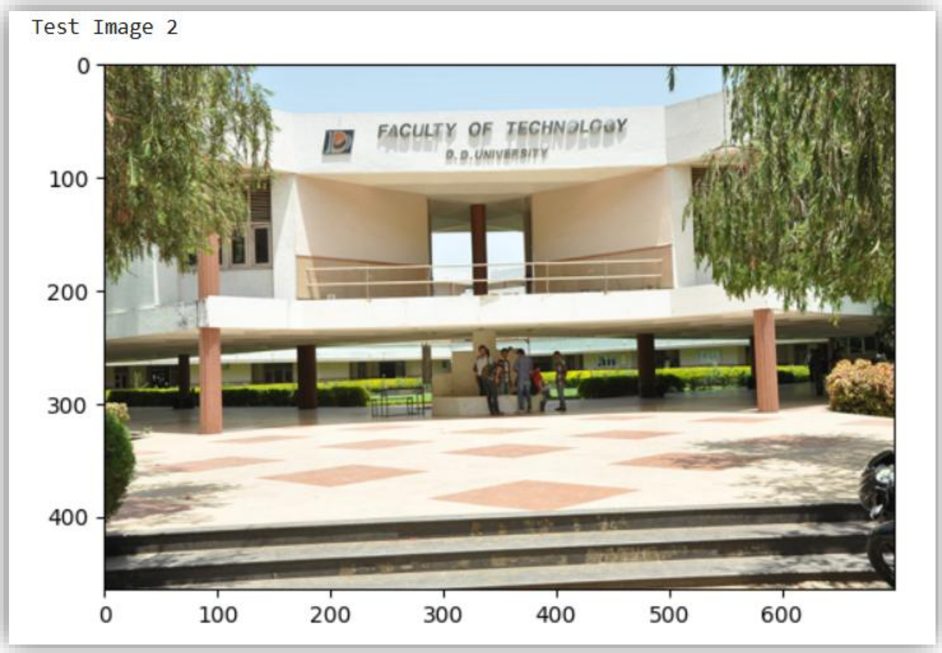
```
print(f"\n\nEasyOCR Results For Test Image 1:\n{easyocr_df_1[['text',
'conf']]}")
print(f"Time taken: {easyocr_time_1:.2f} seconds")
easyocr_df_2, easyocr_time_2 = extract_with_easyocr(image_path_2)
print(f"\n\nEasyOCR Results For Test Image 2:\n{easyocr_df_2[['text',
'conf']]}")
print(f"Time taken: {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 :-

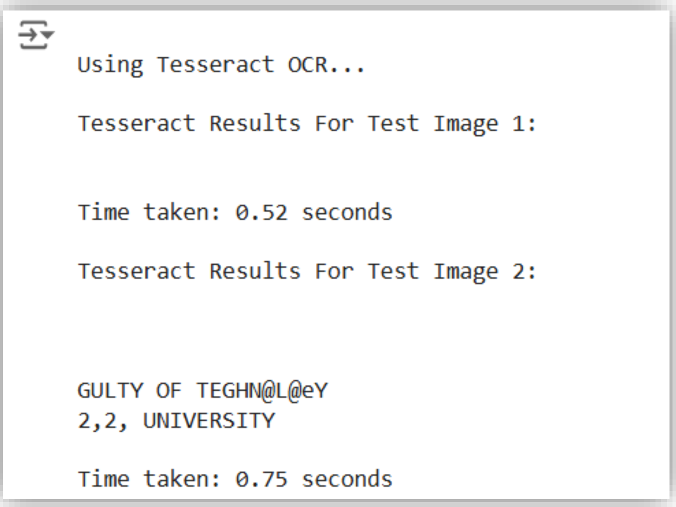
Input Image 1 :-



Input Image 2 :-

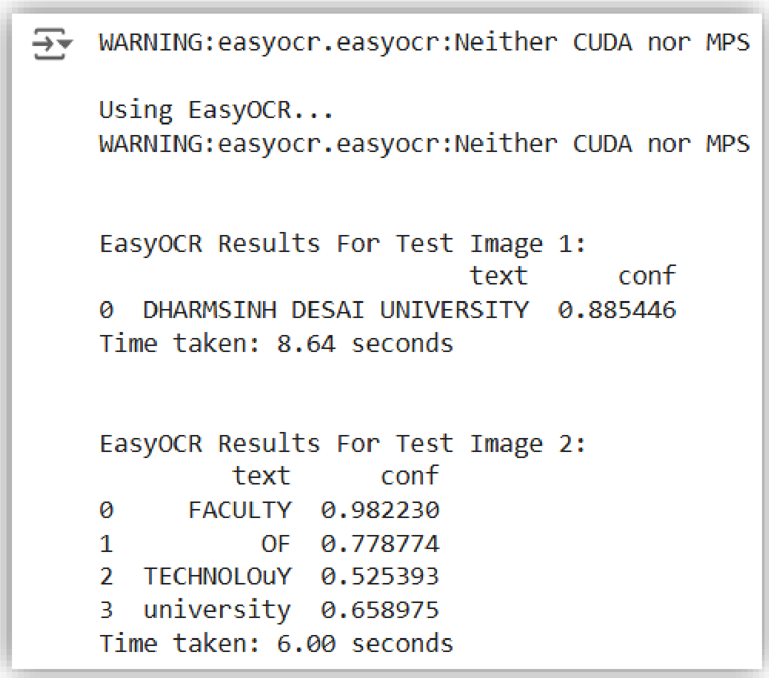


Output Of Tesseract :-



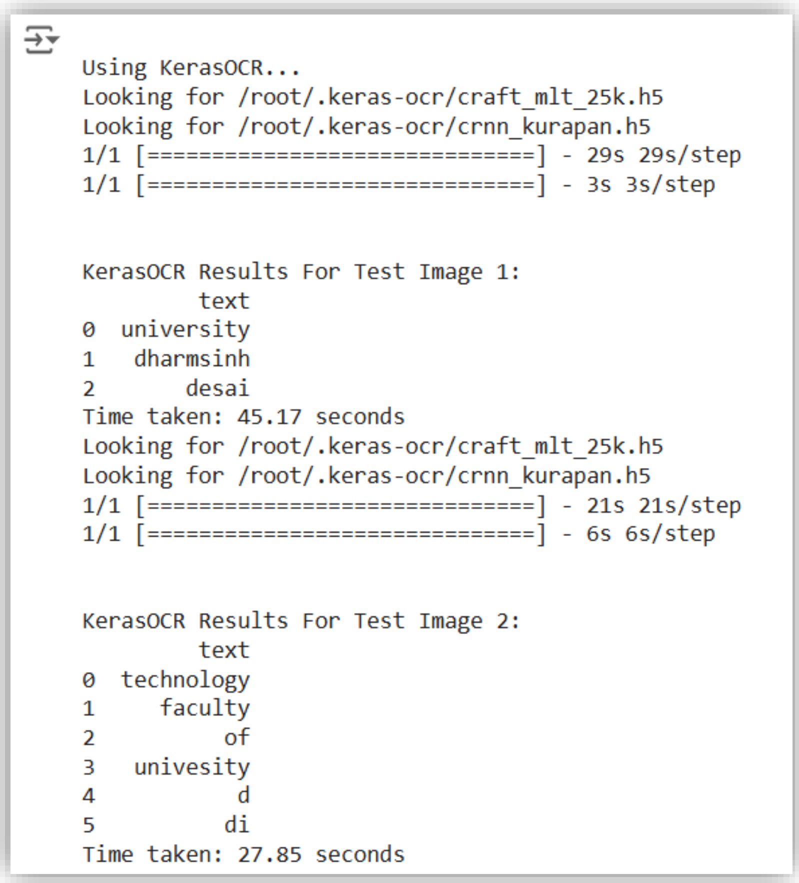
```
Using Tesseract OCR...

Tesseract Results For Test Image 1:


Time taken: 0.52 seconds

Tesseract Results For Test Image 2:



GULTY OF TEGHN@L@eY
2,2, UNIVERSITY

Time taken: 0.75 seconds
```

Output Of EasyOCR :-



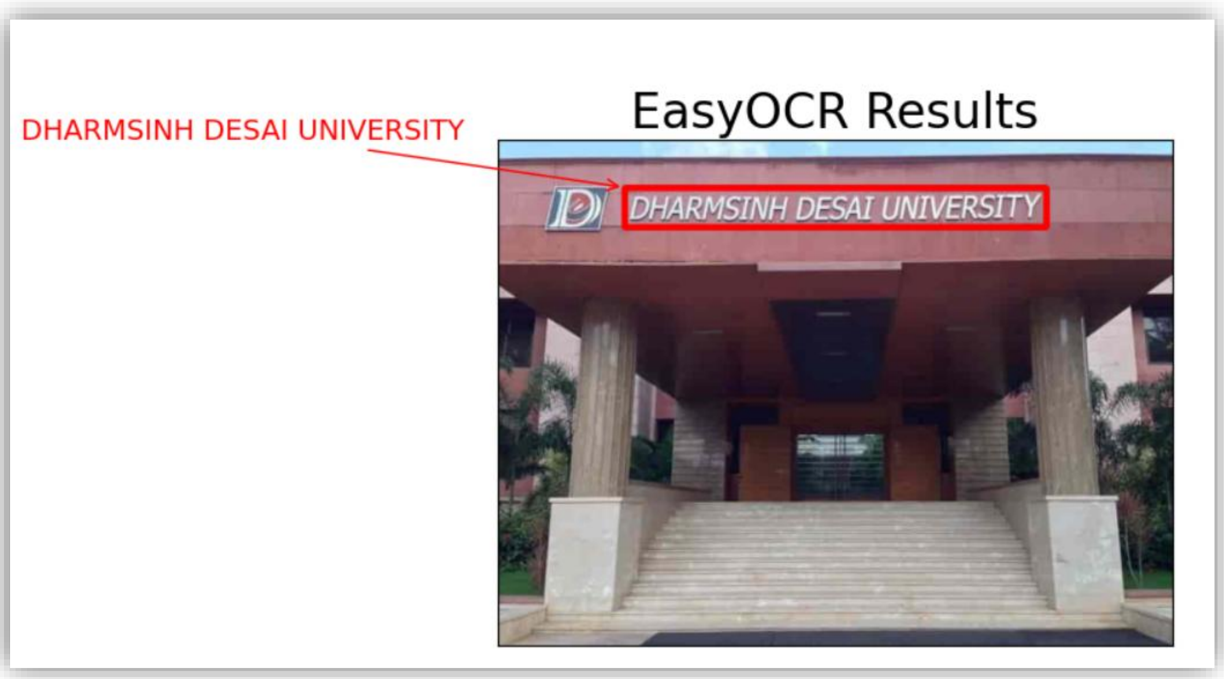
```
WARNING:easyocr.easyocr:Neither CUDA nor MPS

Using EasyOCR...
WARNING:easyocr.easyocr:Neither CUDA nor MPS


EasyOCR Results For Test Image 1:
                        text       conf
0  DHARMSINH DESAI UNIVERSITY  0.885446
Time taken: 8.64 seconds


EasyOCR Results For Test Image 2:
         text       conf
0     FACULTY   0.982230
1          OF   0.778774
2  TECHNOLOuY   0.525393
3  university   0.658975
Time taken: 6.00 seconds
```

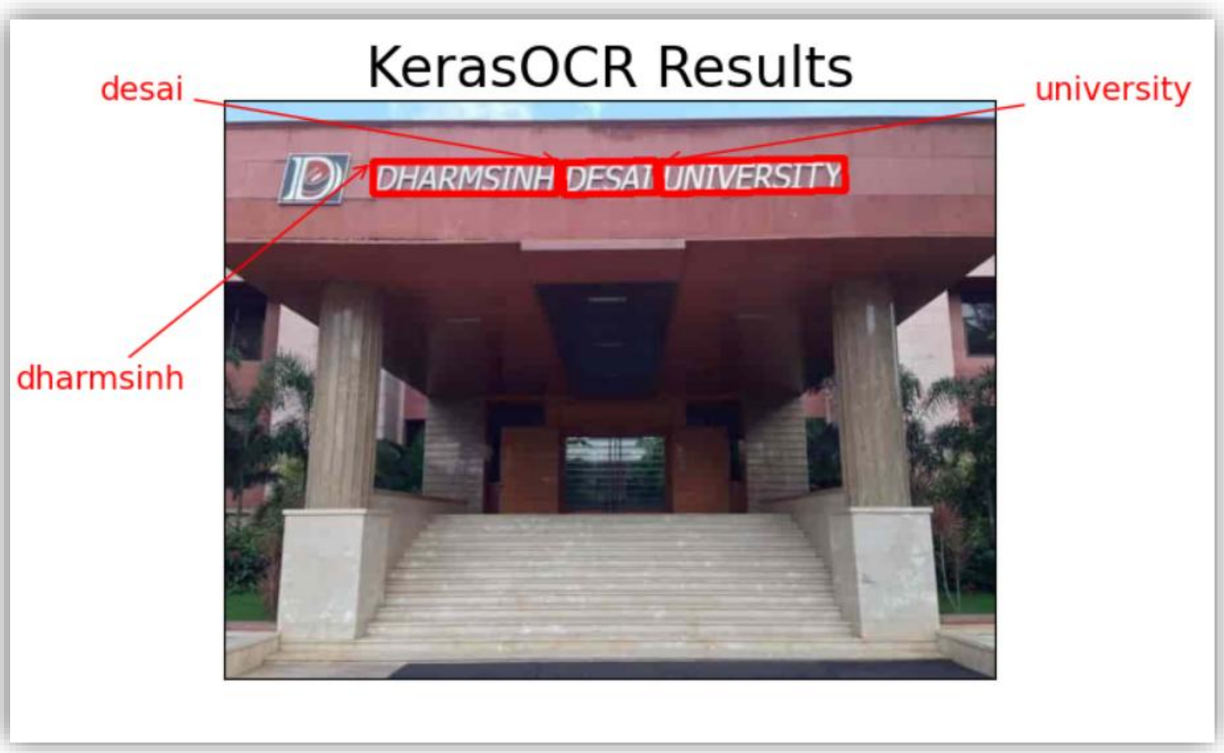
Output Of Keras OCR :-

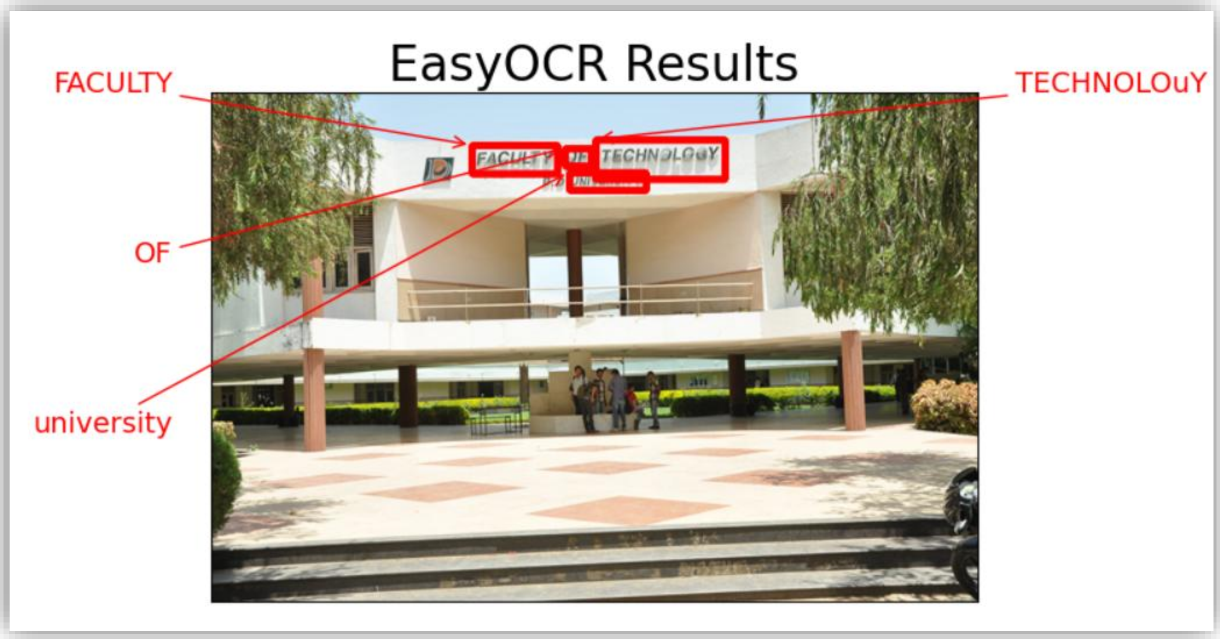


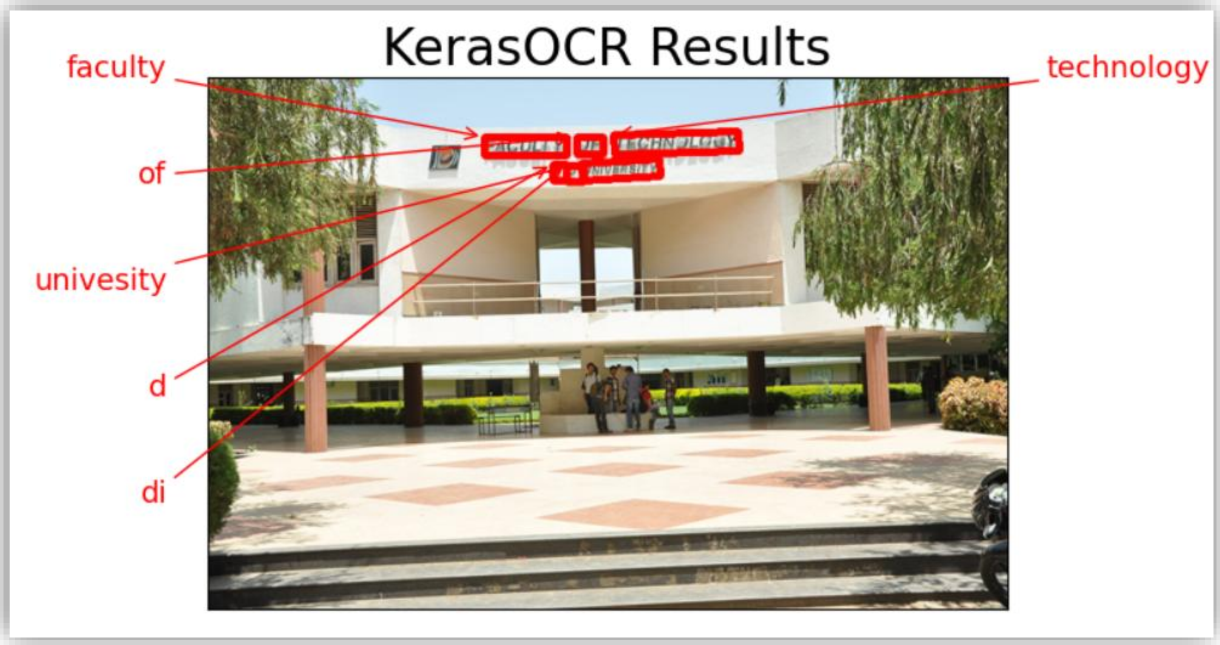
```
Using KerasOCR...
Looking for /root/.keras-ocr/craft_mlt_25k.h5
Looking for /root/.keras-ocr/crnn_kurapan.h5
1/1 [==============================] - 29s 29s/step
1/1 [==============================] - 3s 3s/step


KerasOCR Results For Test Image 1:
         text
0  university
1   dharmsinh
2       desai
Time taken: 45.17 seconds
Looking for /root/.keras-ocr/craft_mlt_25k.h5
Looking for /root/.keras-ocr/crnn_kurapan.h5
1/1 [==============================] - 21s 21s/step
1/1 [==============================] - 6s 6s/step


KerasOCR Results For Test Image 2:
         text
0  technology
1     faculty
2          of
3   univesity
4           d
5          di
Time taken: 27.85 seconds
```

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.