Vivekanand Education Society's

Institute of Technology

(Affiliated to University of Mumbai, Approved by AICTE & Recognized by Govt. of Maharashtra)

Department of Information Technology

AIDS - 2 Lab

Experiment - 4

Aim: To build a Cognitive based application to acquire knowledge through images for a Customer service application/ Insurance/ Healthcare Application/ Smarter Cities/ Government etc.

Roll No.	70
Name	MAYURI SHRIDATTA YERANDE
Class	D20B
Subject	AIDS - 2
Grade:	

EXPERIMENT - 4

AIM: To build a Cognitive based application to acquire knowledge through images for a Customer service application/ Insurance/ Healthcare Application/ Smarter Cities/ Government etc.

THEORY:

A cognitive-based application refers to a type of software or system that leverages artificial intelligence (AI) and cognitive computing techniques to mimic human thought processes and perform tasks that involve understanding, learning, reasoning, and problem-solving.
It can acquire knowledge through images and could utilize computer vision and machine learning techniques to understand, analyze, and learn from visual data.

Handwritten text extraction from images involves using computer vision and optical character recognition (OCR) techniques to convert handwritten text present in images into machine-readable text.
Here's how the process generally works:

1. **Image Preprocessing:** The input image is preprocessed to enhance its quality and prepare it for text extraction. This might involve operations such as resizing, noise reduction, contrast adjustment, and binarization (converting the image to black and white).

2. **Text Localization:** Object detection techniques can be used to identify regions in the image where text is present. This step localizes areas of interest, which could contain handwritten text.

3. **Text Segmentation:** If there are multiple lines or paragraphs of handwritten text, the image might be segmented into individual lines or words to improve the accuracy of text extraction.

4. **Character Recognition:** Optical Character Recognition (OCR) algorithms are applied to recognize individual characters within the segmented regions. For handwritten text, specialized OCR models that handle cursive and diverse handwriting styles might be used. Recurrent neural networks (RNNs) and convolutional neural networks (CNNs) are often used in OCR systems.

5. **Post-processing:** The recognized characters are often subject to post-processing steps to improve accuracy. This could involve correcting misrecognized characters, handling ligatures, and resolving ambiguities.

6. **Language Modeling:** Depending on the language of the text, the OCR system might use language models to improve recognition accuracy. These models consider the probability of certain characters or words occurring based on their context within a sentence.

7. **Text Reconstruction:** The recognized characters are combined to reconstruct words, lines, and paragraphs of text. Contextual information is used to arrange characters in the correct order.

8. **Output Formatting:** The extracted text is often provided in a structured format, such as plain text or a digital document format (PDF, Word, etc.).

IMPLEMENTATION:

- **Extracting text from images and analyzing the sentiment**

(The first line, !pip install easy ocr --quiet, installs the EasyOCR library using the pip package manager. The --quiet flag tells pip to suppress output, so that you don't see a lot of text scrolling by. The second line, import easyocr, imports the EasyOCR library into your Python environment. This allows you to use the functions and classes in the library.)

```
[10] !pip install easyocr --quiet
     import easyocr
     import cv2
     from matplotlib import pyplot as plt
     import numpy as np
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.9/2.9 MB 10.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 813.9/813.9 kB 16.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 146.0/146.0 kB 12.7 MB/s eta 0:00:00
```

```
[11] path = 'hello.jpeg'
```

(The reader variable is an instance of the Reader class from the EasyOCR library. The Reader class is used to read text from images. The readtext() method of the Reader class takes a path to an image file as input and returns a list of strings, where each string is a line of text detected in the image.)

```
[14] reader = easyocr.Reader(['en'])
     result = reader.readtext(path)
     result
```

```
WARNING:easyocr.easyocr:Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.
[([[567, 411], [1185, 411], [1185, 549], [567, 549]],
  'HELLO',
  0.6975526941846563)]
```

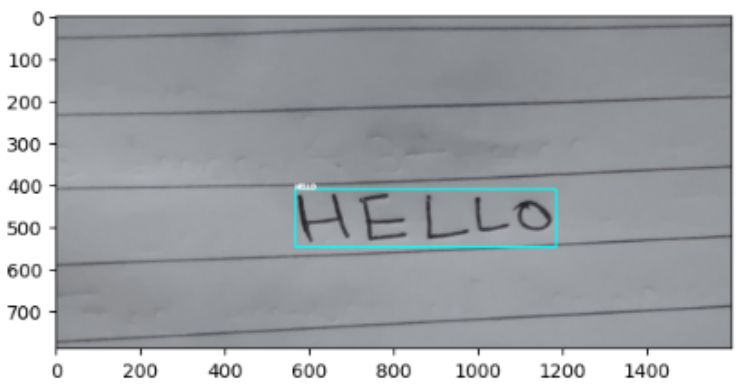
- **Reading the image and Optical character recognition**

(The **top_left variable** is a tuple of two integers, which represent the x and y coordinates of the top-left corner of the bounding box. The **bottom_right variable** is a tuple of two integers, which represent the x and y coordinates of the bottom-right corner of the bounding box.)

```
[15] top_left = tuple(result[0][0][0])
     bottom_right = tuple(result[0][0][2])
     text = result[0][1]
     font = cv2.FONT_HERSHEY_SIMPLEX
```

- **The result gives where the text is in our image and the text which has been recognized and lastly the confidence. Now visualizing where the text is in the image**

```
img = cv2.imread(path)
img = cv2.rectangle(img, top_left, bottom_right, (0, 255, 255), 3)
img = cv2.putText(img, text, top_left, font, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
plt.imshow(img)
plt.show()
```



```
[17] single_text = result[0][1]
     single_text

     'HELLO'
```

- **Finally reading an image with multiple lines of text**

```
[18] path2 = 'welcome.jpeg'
```

```
reader = easyocr.Reader(['en'])
result = reader.readtext(path2)
result
```

```
WARNING:easyocr.easyocr:Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.
[([[431, 107], [692, 107], [692, 210], [431, 210]], 'How', 0.6742406487464905),
 ([[778, 82], [993, 82], [993, 183], [778, 183]], 'Are', 0.6285923771873274),
 ([[1051, 71], [1239, 71], [1239, 159], [1051, 159]],
  'You',
  0.6553024220232424),
 ([[471, 231], [573, 231], [573, 315], [471, 315]], 'I', 0.815418252414247),
 ([[620, 210], [787, 210], [787, 312], [620, 312]], 'AM', 0.9934669317219695),
 ([[821, 192], [1089, 192], [1089, 292], [821, 292]],
  'FINE',
  0.5310249924659729)]
```

```
result

final_text1 = result[0][1]
final_text1
```

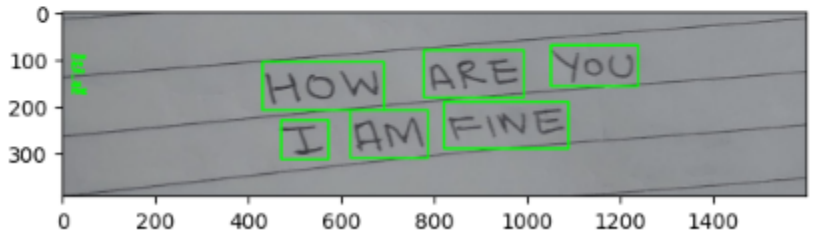
```
'How'
```

```
[24]
final_text1 = result[1][1]
final_text1

'Are'
```

```
[26]
final_text1 = result[2][1]
final_text1

'You'
```

- **Getting the text extracted from the image**

```
[27] img = cv2.imread(path2)
     spacer = 100
     for detection in result:
         top_left = tuple([int(val) for val in detection[0][0]])
         bottom_right = tuple([int(val) for val in detection[0][2]])
         text = detection[1]
         img = cv2.rectangle(img, top_left,bottom_right, (0, 255, 0), 3)
         img = cv2.putText(img, text, (20, spacer), font, 0.5, (0, 255, 0), 2, cv2.LINE_AA)
         spacer += 15
     plt.imshow(img)
     plt.show()
```



- **Sentiment Analysis on the extracted text**

```
!pip install textblob
!python -m textblob.download_corpora
from textblob import TextBlob
import nltk
```

```
Requirement already satisfied: textblob in /usr/local/lib/python3.10/dist-packages (0.17.1)
Requirement already satisfied: nltk>=3.1 in /usr/local/lib/python3.10/dist-packages (from textblob) (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk>=3.1->textblob) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk>=3.1->textblob) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk>=3.1->textblob) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk>=3.1->textblob) (4.66.1)
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Unzipping corpora/brown.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package conll2000 to /root/nltk_data...
[nltk_data]   Unzipping corpora/conll2000.zip.
[nltk_data] Downloading package movie_reviews to /root/nltk_data...
[nltk_data]   Unzipping corpora/movie_reviews.zip.
Finished.
```

- ➢ .The TextBlob object is a Python library that can be used to analyze text. It can identify the parts of speech of words in a text, as well as extract noun phrases.
- ➢ In this case, the tags property will return a list of tuples, where each tuple contains the word and its part of speech. For example, the first tuple in the list ("Hello", "NN"), which means that the word "Hello" is a Noun.
- ➢ The noun_phrases property will return a list of noun phrases in the text. For example, the first noun phrase in the list ("Hello"), which is a noun phrase that consists of the word "hello"

```
blob1 = TextBlob(single_text)
blob1.tags
```

```
[('HELLO', 'NN')]
```

```
[30] blob1.noun_phrases
```

```
WordList(['hello'])
```

```
[31] print(blob1.sentiment)
```

```
Sentiment(polarity=0.0, subjectivity=0.0)
```

- **Finally we perform the sentiment of the second image with multiple words**

```
[33] blob = TextBlob(final_text1)
     blob.tags
```

```
[('You', 'PRP')]
```

```
[34] print(blob.sentiment)
```

```
Sentiment(polarity=0.0, subjectivity=0.0)
```

A polarity of 0.0 means that the text contains an equal number of positive and negative words. A subjectivity of 0.0 means that the text is objective and does not express any personal opinions or beliefs.

➢ The output Sentiment(polarity=0.0, subjectivity=0.0) indicates that the text is neutral, both in terms of its polarity (whether it is positive, negative, or neutral) and its subjectivity (whether it is objective or subjective).

```
[37]
final_text1 = result[5][1]
final_text1
```

```
'FINE'
```

```
[38] blob = TextBlob(final_text1)
blob.tags
```

```
[('FINE', 'NNS')]
```

```
print(blob.sentiment)
```

```
Sentiment(polarity=0.4166666666666667, subjectivity=0.5)
```

➢ Here the polarity of 0.4166666666666667 means that the text contains slightly more positive words than negative words. A subjectivity of 0.5 means that the text is somewhat subjective and expresses some personal opinions or beliefs.

CONCLUSION: OCR software processes the image, identifies characters, and converts them into editable text. This is helpful for digitizing handwritten content, making it searchable and editable. Keep in mind that OCR accuracy can vary based on handwriting quality and the software being used. We also performed Sentimental Analysis on this. Thus we successfully implemented Extracting handwriting text from an image can be achieved using Optical Character Recognition (OCR) technology.