A
PROJECT REPORT
ON

Optical Character Recognition (OCR)

By

SHINGALA SHUBHAM P. (CE-146) (19CEUOS159)
UNAGAR KEVAL V. (CE-167) (19CEUBG010)
VAGHANI SMIT D. (CE-169) (19CEUEG022)

B.Tech CE Semester - VI
Subject : SDP

Guided by:

Prof. Malay Bhatt

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

CERTIFICATE

This is to certify that the practical / term work carried out in subject Of **SDP** and recorded in this journal is the Bonafide work of

SHINGALA SHUBHAM P. (CE 146) (19CEUOS159)

UNAGAR KEVAL V. (CE 167) (19CEUBG010)

VAGHANI SMIT D. (CE 169) (19CEUEG022)

Of B.Tech semester **VI** in the branch of **Computer Engineering**

During the academic year **2021-22.**

Prof. Malay Bhatt
Assistant Professor
Dept. of Computer Engg.,
Faculty of Technology
Dharmsinh Desai University, Nadiad

Dr. C. K. Bhensdadia,
Head,
Dept. of Computer Engg.,
Faculty of Technology
Dharmsinh Desai University, Nadiad

Contents:

1. Software Requirement Specification…… 4
2. Analysis & Design……... 5
3. Design……... 13
   3.1 Data flow Diagram……. 13
4. Implementation Details…….. 14
5. Screen-shots…… 19
6. Conclusion …….. 21
7. Limitation and Future Extension……..... 21
8. Reference / Bibliography……... 22

1.Software Requirement Specification

➢ Purpose of Project :

In today's world, digitization is becoming increasingly important. People typically prefer digitized content to printed items such as books and newspapers, thanks to the expansion of information and communication technology (ICT) and the widespread availability of mobile devices. With sophisticated technology such as artificial intelligence, it is also easier to organize digitized data and analyze it for numerous reasons. So, in order to stay up with the current technological landscape, all of the material now available in printed format must be converted to digitized format.

➢ Scope of Project:

1) Extract Text from Printed English Text Image
2) Works for Following Categories.
   (A – Z, a – z, 0 – 9, Special Characters).

❖ Font Style:

- Calibri
- Consolas
- Ocr A Extended
- Source Code Pro
- Anonymous Pro
- MS Reference Sans Serif
- Yu Gothic
- Cutive Mono
- Lucida Console
- Apercu Mono

❖ Font Size : Minimum 14

❖ Font Background Must be White.

➢ Functional Requirements :

R1: Extract text from Image :

Description: here user will have to upload image of printed text and system will generate extracted text from the image.

Input: upload image

Output: extracted text

Output Format: Word, PDF, TXT, Copy to Clipboard

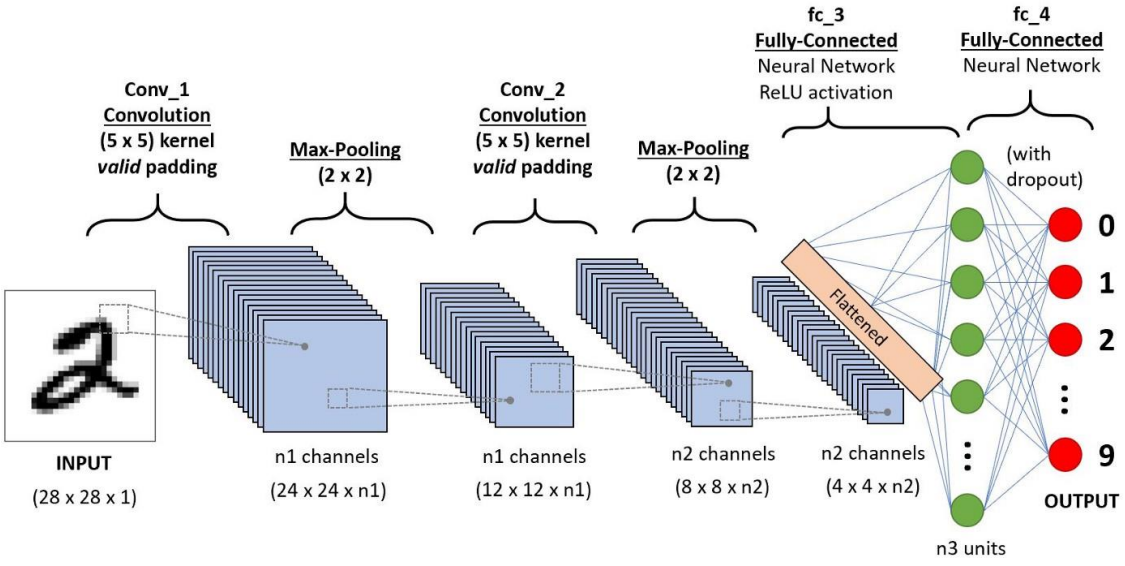
2. Analysis & Design

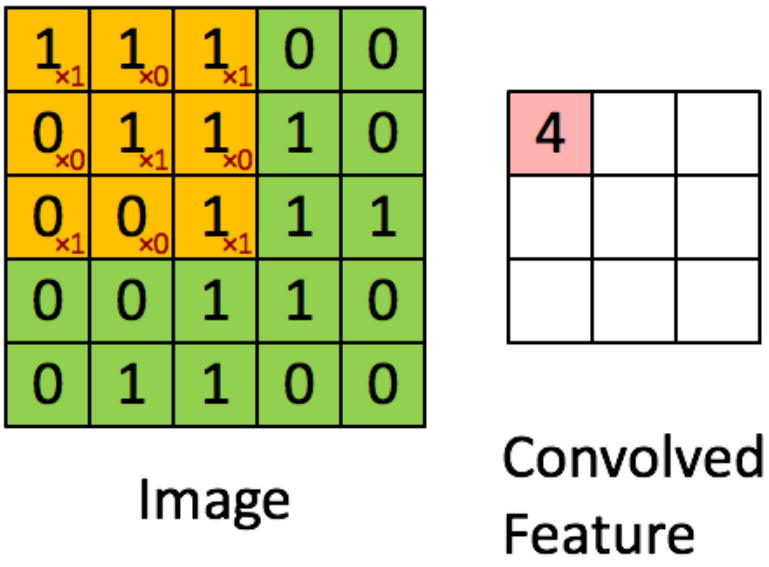
➢ Dataset :

Our Dataset Contains Images of 10 Different font style. Each Image Dimension is 64 x 64. For Training Model, we have converted our image dataset to CSV format.
Training Dataset : 80%
Testing Dataset : 20%

➢ CNN (Convolutional Neural Network) :

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.





In the above demonstration, the green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the colour yellow. We have selected K as a 3x3x1 matrix.

Kernel/Filter, K =

1 0 1
0 1 0
1 0 1

The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering.

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, colour, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation.
1) Valid Padding
2) Same Padding

VALID Padding: it means no padding and it assumes that all the dimensions are valid so that the input image gets fully covered by a filter and the stride specified by you.

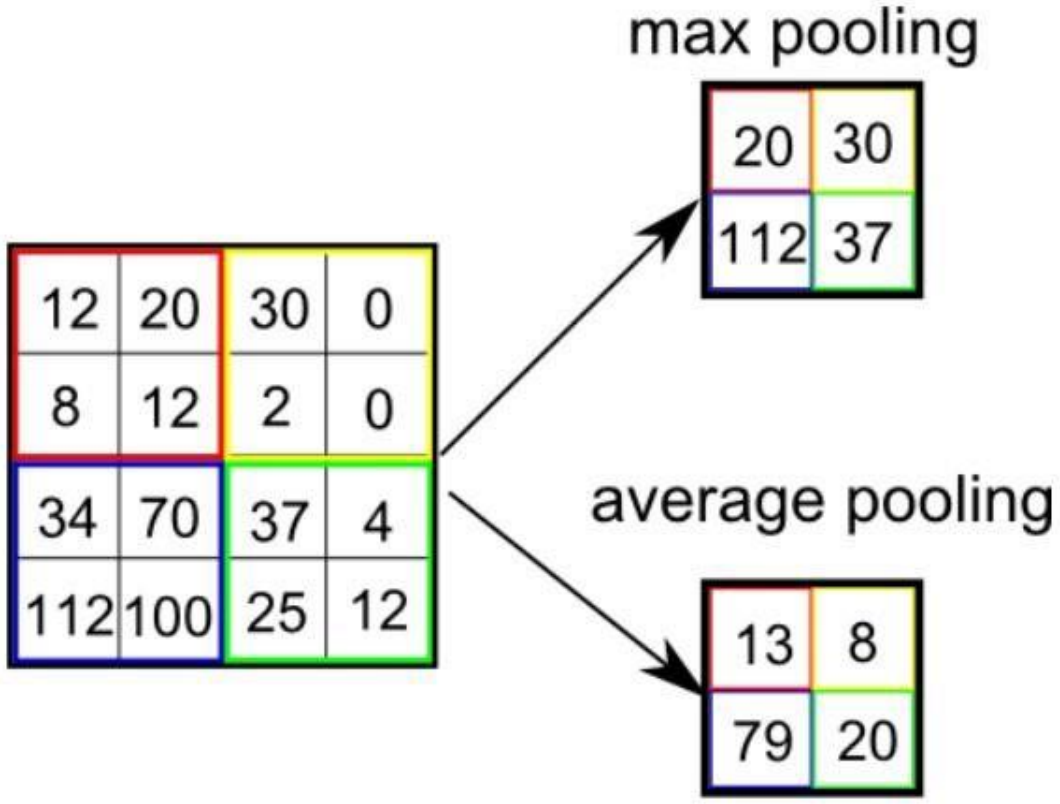
SAME Padding: it applies padding to the input image so that the input image gets fully covered by the filter and specified stride. It is called SAME because, for stride 1 , the output will be the same as the input.

Pooling Layer

There are two types of Pooling:
1) Max Pooling
2)Average Pooling.

Max Pooling: returns the maximum value from the portion of the image covered by the Kernel.

Average Pooling: returns the average of all the values from the portion of the image covered by the Kernel.



After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes. we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

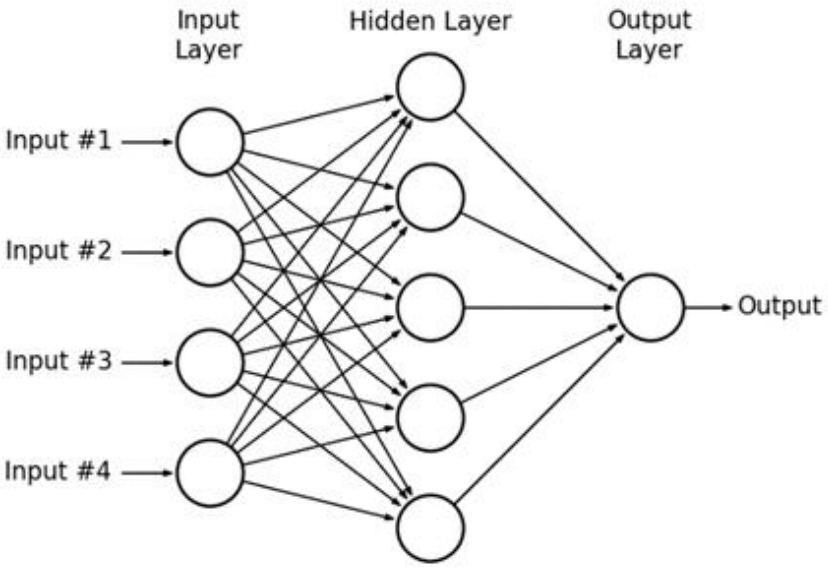
```
data_augmatation=keras.Sequential([
                                    keras.layers.experimental.prepr
ocessing.RandomZoom(0.3)
])
#cnn layers
        data_augmatation,
        keras.layers.Conv2D(filters=32,kernel_size=(3,3),padding='
same',activation='relu',input_shape=(64,64,1)),
        keras.layers.MaxPooling2D((2,2)),
        keras.layers.Conv2D(filters=32,kernel_size=(3,3),padding='
same',activation='relu'),
        keras.layers.MaxPooling2D((2,2)),
        keras.layers.Dropout(0.2),
```

➢ Artificial Neural Network :

Artificial neural networks are one of the main tools used in machine learning. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use.

ANNs are composed of multiple nodes. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value.

Each link is associated with **weight.** ANNs are capable of learning, which takes place by altering weight values. The following illustration shows a simple ANN.



```
#dense layers
        keras.layers.Flatten(),
        keras.layers.Dense(512, activation='relu'),
        keras.layers.Dense(83, activation='softmax')
```

➢ Activation Function

It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 (depending upon the function).

ReLU (Rectified Linear Unit) Activation Function

R(z) = max(0, z)
if z < =0 then R(z) = 0 , if z >0 then R(z) = z

SoftMax Activation Function

SoftMax is an activation function that scales numbers/logits into probabilities. The output of a SoftMax is a vector (say v) with probabilities of each possible outcome. The probabilities in vector v sums to one for all possible outcomes or classes.

➢ Backpropagation

Step 1: Randomly initialize the Weights to a small number close to 0 (but not 0).

Step 2: Input the first observation of your dataset in the input layer, each feature in one input node.

Step 3: Forward-Propagation: from left to right, the neurons are activated in a way that the impact of each neuron's activation is limited by the weights. Propagates the activations until getting the predicted result y.

Step 4: Compare the predicted result to the actual result. Measure the generated error.

Step 5: Back-Propagation: from left to right, the error is back-Propagated. Update the weights according to how much they are responsible for the error. The Learning rate decides how much we update the weights.

Step 6: Repeat step 1 to step 5 and updates the weights after each observation.

Step 7: When the whole training set passed through the ANN that makes an epoch. Redo more epoch.

Error of Model is calculated using cross-entropy.

Log loss = $-y_i * \log(o_{outi}) - (1 - y_i) * \log(1 - o_{outi})$

❖ Steps to build OCR :

- Input Image :

1) Pre-Processing

- Convert Input Image to Gray Scale.

- Threshold (Binarize) Image using (cv2.THRESH_BINARY+cv2.THRESH_OTSU)

2) Segmentation

Method : Projection Profile

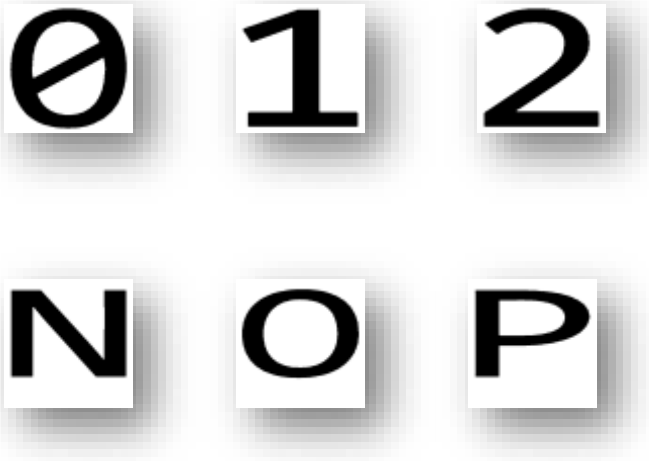
Projection profile is calculated separately for different axis.

Vertical projection: Vertical projection profile is calculated for every column as sum of all row pixel values inside the column.

Horizontal Projection: Horizontal Projection profile is calculated for every row as sum of all column pixel values inside the row.

- Line Segmentation

- Character Segmentation



3) Feature Extraction

The major goal of feature extraction is to extract a set of features, which maximizes the recognition rate with the least amount of elements.

We have divided pixel by pixel image to 255. We have expand channel (Gray scale) to give segmented character image to model.

4) Predict Character using model

⇒ Data Augmentation :

Random Zoom Augmentation The zoom augmentation method is used to zooming the image. This method randomly zooms the image either by zooming in or it adds some pixels around the image to enlarge the image.

Output : **012NOP**

5) Post-Processing

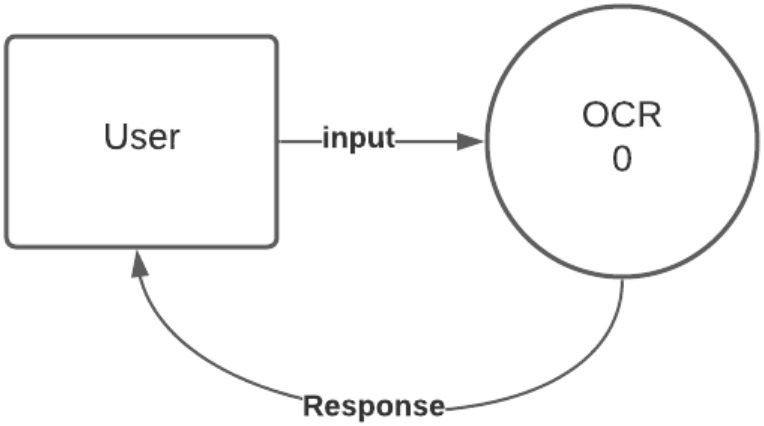
Output of Predicted characters gets combined to make a paragraph by adding space and enter.

Output :

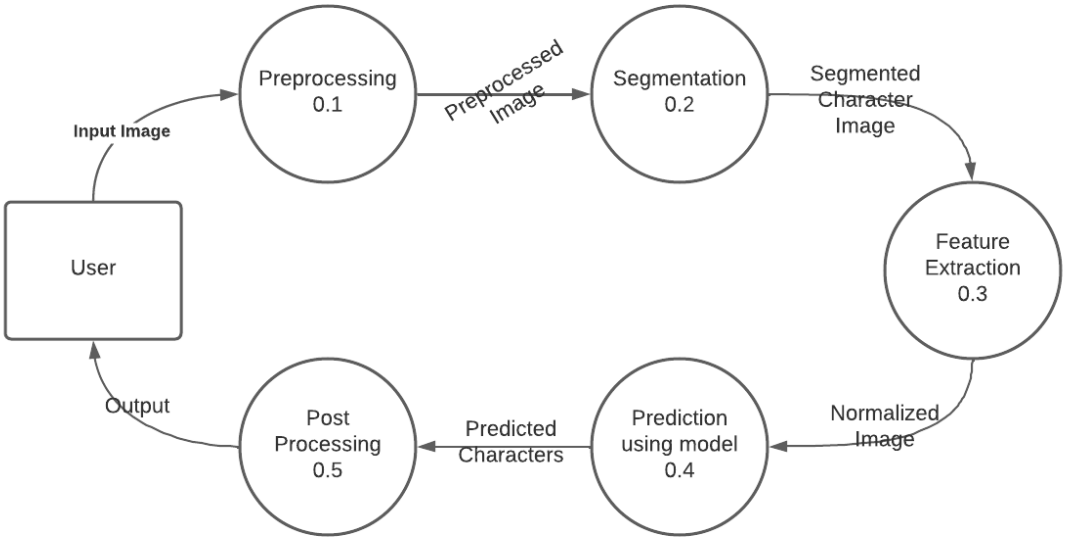
012
NOP

3. Design

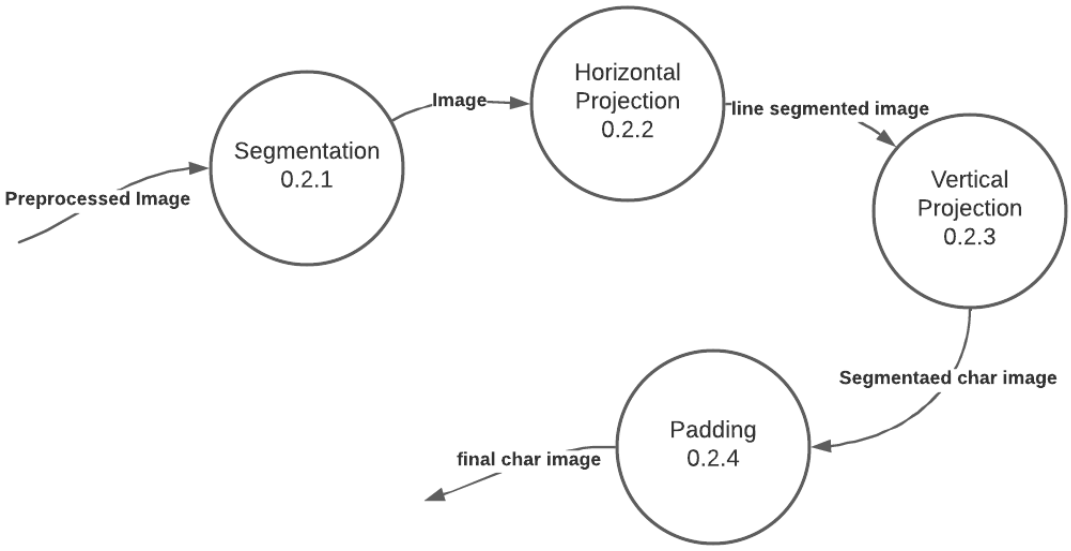
❖ **Data Flow Diagram:**



Level – 0 Context Diagram



Level – 1 Diagram



Level – 2 Diagram

4. Implementation Details

- **OCR Model Implementation:**

⇒ **Import**

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import cv2
import pandas as pd
import matplotlib.pyplot as plt
from keras.preprocessing import image
%matplotlib inline

from cv2 import bitwise_not
from cv2 import INTER_AREA
import os
import imutils

target_labels="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmn
opq
rstuvwxyz!@%&*()+={}[]:;',#.?-"
```

⇒Train Model

```
df=pd.read_csv('/content/drive/MyDrive/data_64_64_10_fonts_dataset.csv',header=None)
df.shape

x=df.drop(0,axis='columns')
x=x.to_numpy()/255
x=x.reshape(x.shape[0],64,64)
x=np.expand_dims(x,axis=-1)
x.shape
y=df[0].to_numpy()
y.shape
plt.imshow(x[4600].reshape(64,64),cmap='gray')
print(y[4600])

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,shuffle=True,random_state=10)

x_train.shape,x_test.shape,y_train.shape,y_test.shape

data_augmatation=keras.Sequential([
 keras.layers.experimental.preprocessing.RandomZoom(0.3)
])
cnn = keras.Sequential([
        #cnn layers
        data_augmatation,

keras.layers.Conv2D(filters=32,kernel_size=(3,3),padding='same',activation='relu',input_shape=(64,64,1)),
        keras.layers.MaxPooling2D((2,2)),

keras.layers.Conv2D(filters=32,kernel_size=(3,3),padding='same',activation='relu'),
        keras.layers.MaxPooling2D((2,2)),
        keras.layers.Dropout(0.2),
        #dense layers
        keras.layers.Flatten(),
        keras.layers.Dense(512, activation='relu'),
        keras.layers.Dense(83, activation='softmax')
    ])
cnn.compile(
    optimizer='adam',
    loss="sparse_categorical_crossentropy",
    metrics=['accuracy']
)
cnn.fit(x_train,y_train,epochs=50)
```

```
outputs = [layer.output for layer in cnn.layers]
outputs

cnn.summary()

loss,accuracy=cnn.evaluate(x_test,y_test)
print("accuracy:",accuracy)

# cnn.evaluate(x_test,y_test)
cnn.save("/content/drive/MyDrive/data_64_64_10_fonts_datasets.h5")

num=426
plt.imshow(x_test[num].reshape(64,64),cmap='gray')
target_labels[np.argmax(cnn.predict(x_test)[num])]
```

⇒Predict Output

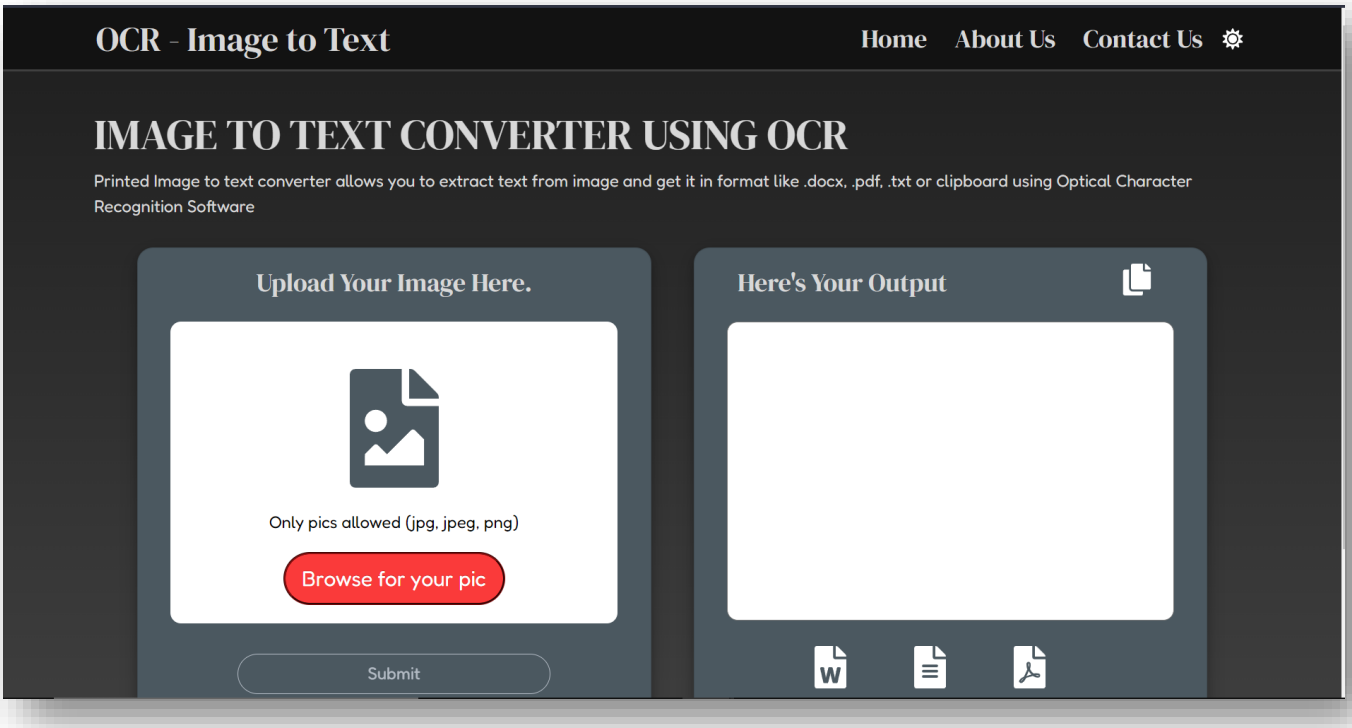
```
cnn=keras.models.load_model('/content/drive/MyDrive/data_64_64_10_fo
nts_datasets.h5')

def showProjection(binary,projection,typeProject="horizental"):
  h,w=binary.shape
  project=np.zeros(binary.shape, dtype=np.uint8)
  if typeProject=="vertical":
    for i in range(w):
        for j in range(projection[i]):
            project[j, i] = 255
  else:
    for j in range(h):
          for i in range(projection[j]):
              project[j, i] = 255
  plt.imshow(project,cmap='gray')

# Horizontal projection
def hProject(binary):
    h,w=binary.shape
    temp=binary.copy()
    temp[temp==0]=1
    temp[temp==255]=0
    h_projection=np.sum(temp,axis=1)
    # showProjection(binary,h_projection)
    return h_projection

# Vertical back projection
def vProject(binary):
    h,w=binary.shape
    temp=binary.copy()
    temp[temp==0]=1
```

```
    temp[temp==255]=0
    v_projection=np.sum(temp,axis=0)
    # showProjection(binary,v_projection,"vertical")
    return v_projection

def charSegmentation(image):  # load image in gray scale
    image = cv2.resize(image, None, fx=4,
fy=4,interpolation=cv2.INTER_CUBIC)
    ret, thresh = cv2.threshold(image, 127,
255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

    th = thresh
    start=0
    h, w = th.shape
    h_h = hProject(th)
    h_start, h_end = [], []
    position = []

    # Vertical segmentation based on horizontal projection
    for i in range(len(h_h)):
        if h_h[i] > 0 and start == 0:  # >0 means 1 or more black pixel
            h_start.append(i)
            start = 1
        if h_h[i] == 0 and start == 1:
            h_end.append(i)
            start = 0

    for i in range(len(h_end)):
      cropImg = th[h_start[i]:h_end[i], 0:w]

      lh,lw = cropImg.shape
      if i == 0:
          pass

      w_w = vProject(cropImg)
      wstart, wend, w_start, w_end = 0, 0, 0, 0
      isEnter=False
      for j in range(len(w_w)):

        if i!=0 and j == 0: # not first line and it is first char
          isEnter = True

        if w_w[j] > 0 and wstart == 0:  # >0 means 1 or more black
pixel
          w_start = j
          wstart = 1
          wend = 0

        if w_w[j] == 0 and wstart == 1:
```

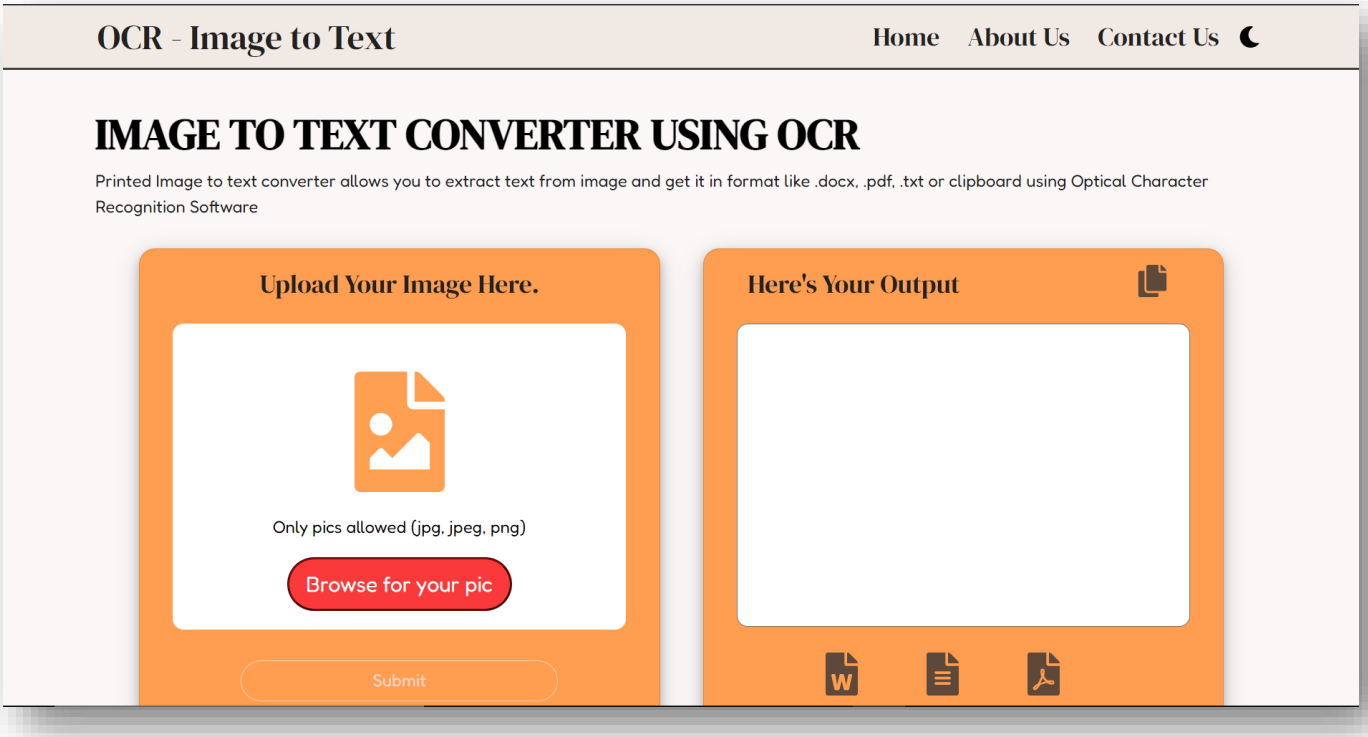
```
          w_end = j
          wstart = 0
          wend = 1

            # Save coordinates when start and end points are confirmed
        if wend == 1:
          isSpace = False
          count = 0
          while(w_w[j]==0):
            count+=1
            j+=1
            if count>lh/4:
              isSpace = True
              break
          position.append([w_start, h_start[i], w_end,
h_end[i],isSpace,isEnter])
          isEnter=False
          wend = 0
    char=''
    roiArr=[]
    space=[]
    enter=[]
    # # Determine division position
    for i,p in enumerate(position):
        roi = thresh[p[1]:p[3], p[0]:p[2]]
        roi = cv2.copyMakeBorder(roi, top=15, bottom=15, left=15,
right=15, borderType=cv2.BORDER_CONSTANT,value=(255,255,255))
        roi = cv2.resize(roi, (64, 64), interpolation=cv2.INTER_AREA)
        roiArr.append(roi)
        if p[5]:
          enter.append(i)

        if p[4]:
          space.append(i)

    roiArr = np.asarray(roiArr)

    for i,element in enumerate(cnn.predict(roiArr)):
      if i in enter:
        char+='\n'
      char+=target_labels[np.argmax(element)]
      if i in space:
        char+=' '
    return char
image=cv2.imread("/content/sample_data/calibri14_test2.png",0)
image = cv2.copyMakeBorder(image, top=15, bottom=15, left=15, right=15,
borderType=cv2.BORDER_CONSTANT,value=(255,255,255))
char=charSegmentation(image)
print(char)
```

5. Screenshots

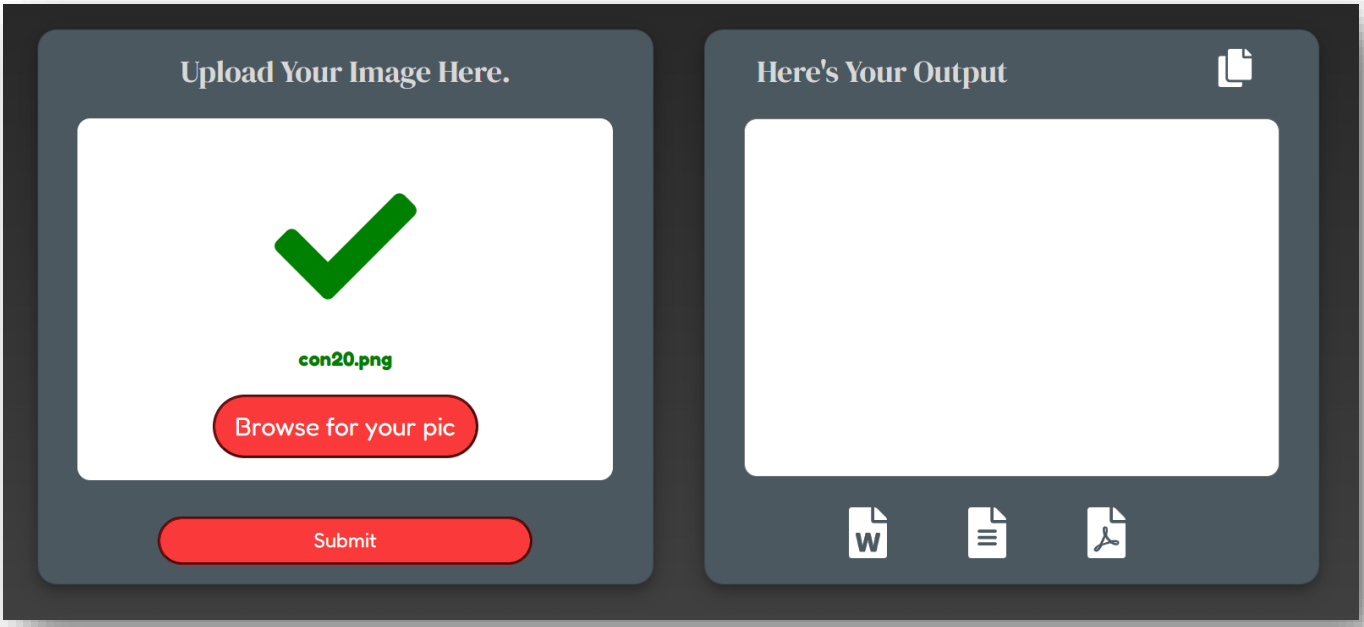
Dark Theme:



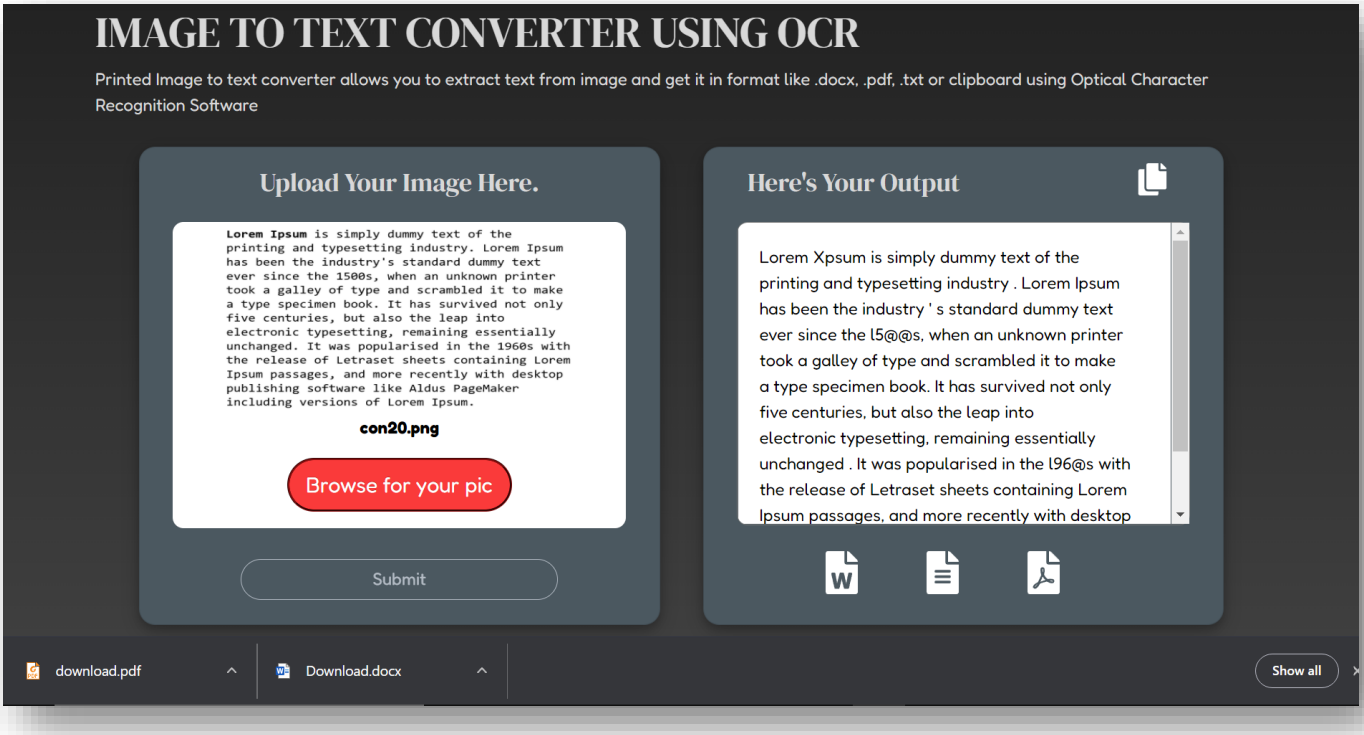
Light Theme:



File Upload:



Output:



6. Conclusion

- Functionality that is Successfully implemented in the system is :
  - Extract text from Image
- Accuracy of our machine learning model of OCR is: 99%

```
loss,accuracy=cnn.evaluate(x_test,y_test)
print("accuracy:",round(accuracy, 2)*100,"%")
```

```
31/31 [==============================] - 3s 88ms/step - loss: 0.0303 - accuracy: 0.9896
accuracy: 99.0 %
```

7. Limitations and Future Enhancements

- As we have used projection profile method for segmentation so, it may not give good result for italic fonts. And it can be solved by using other segmentation methods like contour.
- As we have created our machine learning model using some specific font styles so, it may not give good accuracy for other font styles.
- For future enhancements we can add more font style images in our data sets and re-build model using new data sets as per requirement.
- If font size is less than 14 then, we will not get segmented image properly separated so , it may not give good accuracy.
- If input image is skewed at some angle, then out OCR will not work. And it can be solved by finding angle of rotation and rotate it in reverse direction.

8. Reference / Bibliography

Following links and websites were referred during the development of this

Project:

https://github.com/Shubham-Shingala/OCR-Printed_English_Text.git

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

https://medium.com/machine-learning-researcher/artificial-neural-network-ann-4481fa33d85a

https://flask.palletsprojects.com/en/2.0.x/

https://stackoverflow.com/

https://towardsdatascience.com/what-is-ocr-7d46dc419eb9

https://www.youtube.com/playlist?list=PLeo1K3hjS3uu7CxAacxVndI4bE_o3BDtO