M S RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute Affiliated to VTU)
Department of Information Science and Engineering

Dec 2019

A Report on

Automated marks updation by detection of USN and Marks on the blue book

Submitted in partial fulfillment of the CIE for the subject

DIGITAL IMAGE PROCESSING

Signature of Faculty

Project Team-

SAHANA JP (1MS17IS099)
AMAN BHATNAGAR (1MS17IS144)
SANKETH CS (1MS17IS102)
SANJAY BR (1MS17IS101)

TABLE OF CONTENTS :

ABSTRACT

The aim of this project is to develop an Optical Character Recognition (OCR) for a STUDENT INFORMATION SYSTEM (SIS) which is automatically updated in the database and UI by feeding in a picture of the USN and its respective marks. The project also contains a User Interface to display the same. The image is fed, goes under a series of pre-processing techniques like segmentation, gray scaling and thresholding after which it is fed to a trained machine learning model implemented using Keras and Convolutional Neural Networks to detect the handwritten characters in the scanned image. Our project aims to overcome the usual indirect process of marks updation and provides a direct and easy-to-use alternative which can be used in many institutions and schools.

INTRODUCTION

The traditional way professors use to update a student’s marks is by manually updating it in their Portal. Our project suggests a new way to ease up the process. We apply technology in almost everything we do in our daily lives, so why not here. Thanks to the OCR and Machine learning technologies that facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs. We have used basic image processing techniques like segmentation to process the input image. This image is then fed to a trained machine learning model for prediction of the unknown image. We have made it a point to build a good looking User Interface website which we hosted using Flask framework. Due to time and system configuration constraints, the output accuracy we obtained was not up to the mark, so giving feasibility the utmost priority we also used the Google Cloud Vision API for this process, and the results we obtained were way accurate. We are living in an era of “**Move fast and break things**” (Mark Zuckerberg’s famous motto), and we tried our best to implement technology for the better.

DESIGN / ALGORITHM

We used a Convolutional Neural Network (CNN Algorithm) available as a part of Keras, python’s machine learning library and used OpenCV for Image Segmentation and Pre-processing techniques.

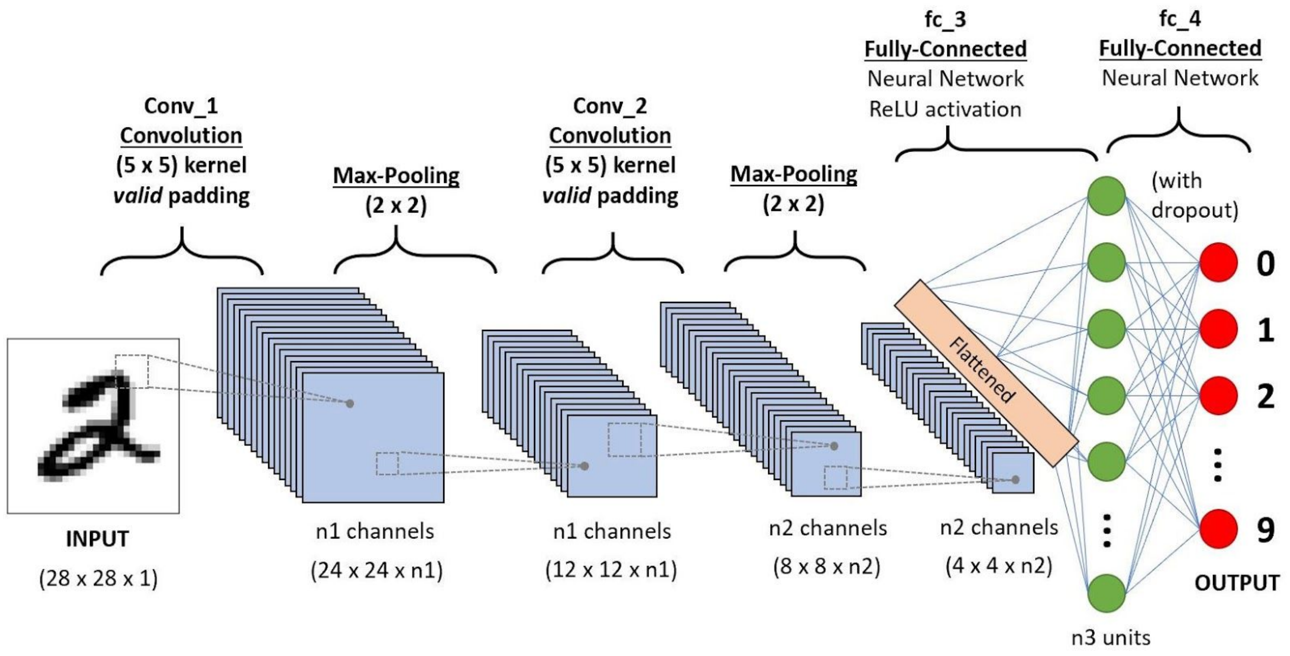
Image Preprocessing included -

- Reading
- Gray Scaling
- Thresholding
- Contours detection
- Determining the height width and required axis

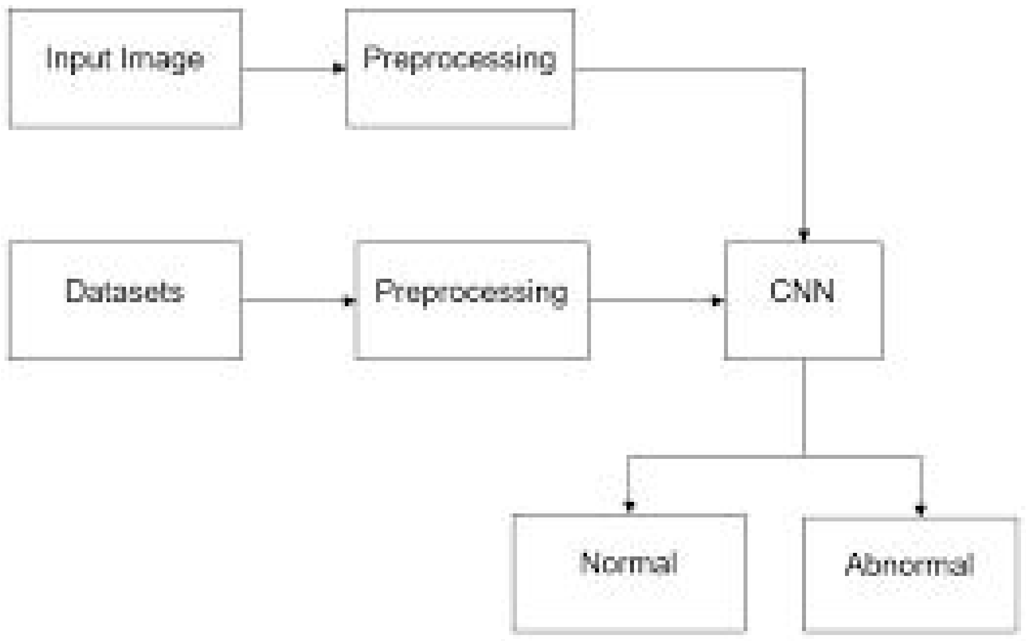
The neural network has 5 layers -

- Convolutional Layer
- Pooling Layer
- Flattening
- Relu Activation
- Fully Connected Neural Network

The layer diagram is as follows :



The Block Diagram is as follows :



IMPLEMENTATION

The Code is as follows :

Model File :

```
# importing the
dependencies
import os
from os import listdir
from os.path import isfile, join
import cv2
import numpy as np
import keras
from keras import backend as k
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers.core import Flatten, Dense




# a function to resize the image into appropriate dimensions
def resize(img):
   img = cv2.resize(img,(20,20))
   return img


X_train = []
y_train = []
```

```
# to get the name of the folder
for name_folder in os.listdir("extracted_letter_images") :


   name = 'extracted_letter_images/' + name_folder
   for f in listdir(name):
       # name of the folder is the name of the output
       y_train.append(np.asarray(name_folder))


       # constructing full path to the image
       name = 'extracted_letter_images/' + name_folder + '/' +
f


       # reading the image
       image = cv2.imread(name,0)/255


       # appending to form the image list
       image = np.asarray(image)
       image = resize(image)
       X_train.append([image])


# converting the lsit into an numoy array so that it can be
fed into neural network
X_train = np.asarray(X_train)
y_train = np.asarray(y_train)


X_train = np.reshape(X_train, [-1,20,20,1])




X_train.shape
```

```
# one hot encoding the output labels
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(y_train)

onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded =
integer_encoded.reshape(len(integer_encoded), 1)
y_train = onehot_encoder.fit_transform(integer_encoded)

# defining the architecture of the model
model = Sequential()

# First convolutional layer with max pooling
model.add(Conv2D(64, (3, 3), padding="same",
input_shape=(20,20,1), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Second convolutional layer with max pooling
model.add(Conv2D(128, (3, 3), padding="same",
activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

# third convolutional layer with max pooling
model.add(Conv2D(512, (3, 3), padding="same",
activation="relu"))
```

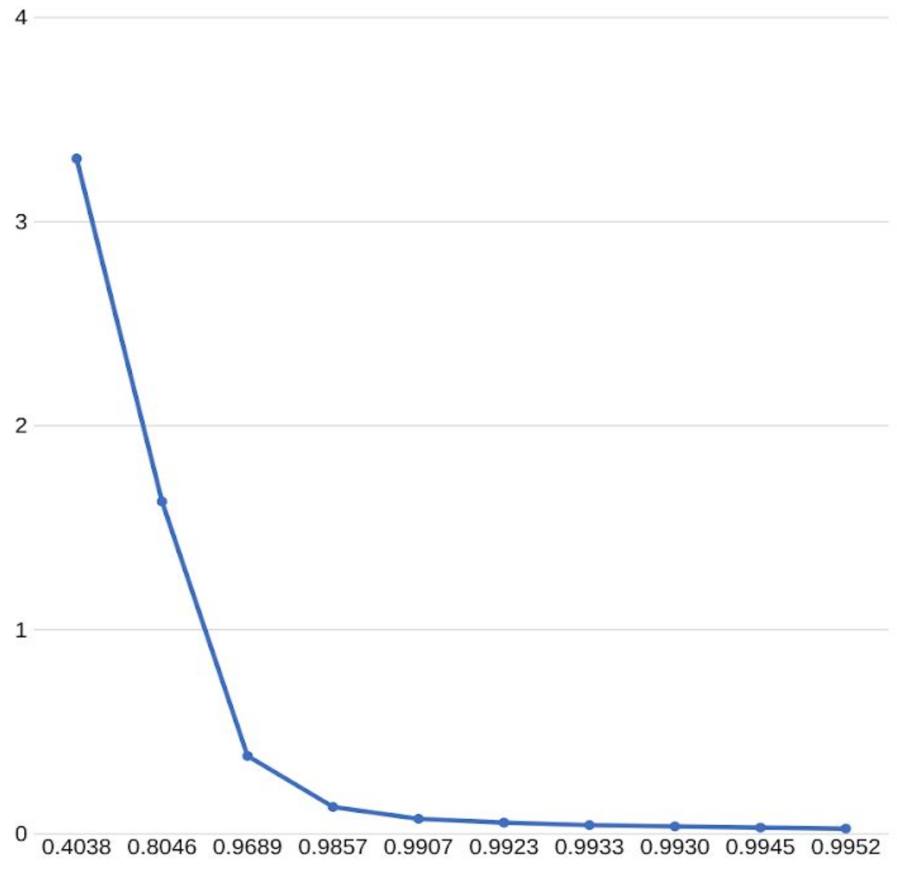
```
model.add(MaxPooling2D(pool_size=(2, 2)))


# Hidden layer with 500 nodes

model.add(Flatten())

model.add(Dense(100, activation="relu"))


# Output layer with 32 nodes (one for each possible
letter/number we predict)
model.add(Dense(32, activation="softmax"))


# Ask Keras to build the TensorFlow model behind the scenes

model.compile(loss="categorical_crossentropy",
optimizer="adam", metrics=["accuracy"])


# splitting the samples into training and testing sets so that
we cross validate our results
X_train, X_test, y_train, y_test =
train_test_split(X_train,y_train ,test_size = .2)


# shape of the training output

y_train.shape


# fitting the model

model.fit(X_train, y_train,

         validation_data=[X_test,y_test],

         batch_size=2000,

         epochs=10,

         verbose=1)
```

```
#saving the model
model.save('models/model.h5')


np.save('models/xtrain',X_train,allow_pickle=True)
```

Prediction File :

```
# importing
dependencies
import numpy as np
import keras
import os
import cv2
import imutils
import os.path
import importlib
from os import listdir
from keras import backend as k
from keras.models import load_model
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder


y_train = []
# to get the name of the folder
for name_folder in os.listdir("extracted_letter_images") :
   name = 'extracted_letter_images/' + name_folder
   for f in listdir(name):
       # name of the folder is the name of the output
       y_train.append(np.asarray(name_folder))
y_train = np.asarray(y_train)
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(y_train)



def image_segmentation(image_name):
   # reading the image
   image = cv2.imread(image_name)

   # converting the image to grayscale
   gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

   # threshold to convert the image to pure black and white
   thresh = cv2.threshold(gray, 0,255, cv2.THRESH_BINARY_INV |
cv2.THRESH_OTSU)[1]


   # find the contours (continous blob of pixels ) in the image
   contours = cv2.findContours(thresh,cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

   # Hack for compatibility with different OpenCV versions
   contours = contours[0] if imutils.is_cv2() else contours[1]

   letter_image_regions = []
   # now loop through each of the letter in the image
   for contour in contours:
       # get the rectangle that contains the contour
       x,y,w,h = cv2.boundingRect(contour)
       # compare the width and height of the contour to detect
if it
       # has one letter or not
       if w/h >1.25:
           # this is too wide for a single letter
           continue
       elif w<3 or h<3:
           # this is a very small image probably a noise
           continue
       else:
       # this is a normal letter by itself
           letter_image_regions.append((x,y,w,h))


   return letter_image_regions


   # loading the trained model
model = load_model('models/model.h5')


# a function to resize the image into appropriate dimensions
def resize(img):
   img = cv2.resize(img,(20,20))
   return img


# now we will read images from the folder segment the images
and will produce the output
for image_name in listdir('images'):
    counter = 1
    # constructing the name of the file
    file_name = 'images/' + image_name


    # getting segmented images
    letters_in_image = image_segmentation(file_name)

    # sorting the letters so that letters that appear before is
addressed first
    letters_in_image = sorted(letters_in_image, key=lambda x:
x[0])

    ans = ""
    for (x,y,w,h) in letters_in_image:
        image = cv2.imread(file_name,0)
        letter = image[y - 2:y + h + 2, x - 2:x + w + 2]

        cv2.imwrite(str(counter)+'.jpg', letter)
        counter = counter + 1

        letter  = resize(letter)/255
        X_test = np.asarray(letter)
        X_test = np.reshape(X_test, [-1,20,20,1])
        output = np.argmax(model.predict(X_test, verbose = 0))
        output = label_encoder.inverse_transform(output)
        ans +=output
    print("image no: ", ans)
```

RESULT ANALYSIS



Y - AXIS : Loss

X - AXIS : Accuracy

CONCLUSION

Thus we tried optimising and easing out the job of updating the student information systems in colleges. It can become a very powerful tool for future data entry applications. With given proper imputes and encouragement, a lot of benefits can be provided in this project.The automated entry of data is one of the more attractive ,labor reducing technology. The recognition made is so easy and accurate.

To conclude, we implemented Optical Character Recognition and a full backend setup to successfully render that output in a webpage specifically created for the purpose. The automated Bluebook marks and USN detection and updation programme can access all kinds of texts and picks up different handwriting. It overcomes many limitations and can be implemented in colleges and universities by its Easy Implementation and High Flexibility.

REFERENCES

Youtube - https://www.youtube.com/watch?v=s_ht4AKnWZg
Youtube - https://www.youtube.com/watch?v=tqFk8bzv2ys
Github - https://github.com/amyy28/DIP_Project/blob/master/ML_prediction.py
Links - https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa
Links - https://mc.ai/ocr-part-3%E2%80%8A-%E2%80%8Aocr-using-image-segmentation-and-cnn/