CSCE 5290

Natural Language Processing

INCREMENT -1

GitHub link: https://github.com/nehareddy9399/NLP-Projectgroup11

1.Project Title and Members:

Project Title: Optical Character Recognition using Deep Learning

Members: (Team 11)
Keerthana Pinikeshi (Student ID - 11448714)
Neha Kalluri (Student ID - 11445206)
Lahari Kunduru (Student ID - 11445208)

2.Goals and Objectives:

Motivation:

As part of this NLP project, we thought of building something that is very close to the real world and widely used in NLP applications. We found that Optical Character Recognition (OCR) is one of the most popular areas of research in NLP applications.

We as a team felt that the Optical Character Recognition problem statement will be used by all kinds of individuals in their day-to-day life. Hence we are proposing to work on the Optical Character Recognition model to recognize the text in the given input image.

Significance :

The major significance of Optical Character Recognition (OCR) is used in almost all kinds of industries and individuals. Before the advancements in the OCR, we used to look at the hard copies of documents and type the content manually. But now we can process any document or image to extract the text from the OCR. It saves huge time and manpower in almost every industry and every individual's life.

OCR technology is used very often in real-world applications. Some of the applications of the OCR are listed below.

1.Data entry of documents

2.Data extraction from KYC documents

3.Number plate recognition

4.Self Driving Cars

5.Hand Written text recognition.

6.Scanned documents to Searchable PDFs

7.CAPTCHA recognition

Objectives :

The main objective of our project is to build a real-world OCR model to recognize the text in the image. As we have researched, building an OCR model which recognizes the whole text in the image is not scale-able and required huge computation power along with powerful models and datasets. We have also researched that most of the OCR products perform word-by-word extraction and concatenate the words to get the whole text extraction. Hence we are going to build the OCR model for the extraction of the single word.

In order to build the OCR model, the following are objectives we followed:

- **Data Collection:** We need to do some research and pick the right dataset for this project. We have collected some datasets and we would like to spend some more time on the data collection.

- **Data Cleaning:** It is very important to clean the data after collection. Sometimes there could be wrong annotations in the dataset. More the time we spend on data cleaning, more the quality results we can get.

- **Data Preprocessing:** In this phase, we will take the image and perform some preprocessing techniques like data augmentation, rotation, adding noise, etc. Here, we will convert the image to a multi-dimensional array.

- **Model Building:** Since the dataset is in the image format, it is required to build a deep neural network model instead of machine learning models. We need to build a suitable model architecture.

- **Model Training:** In this section, we will train the model, and also we have tried various hyperparameters like learning rate, batch size, etc.

- **Evaluation of results:** In section, we evaluate the results by checking the loss and plotting the image and its recognized text.

Features :

In general, to train a deep learning model, we require GPU computing power. By using Google collab we can get the free GPU for a limited time per day. Colab's computer power will be sufficient to train the OCR model. Colab also provides the option to mount google drive. Since we are going with a huge dataset, we uploaded the dataset to google drive once so that we can easily able to mount the drive and use it for the model training.

Since we are training the OCR model from scratch, We require a huge dataset. We would like to choose a dataset that contains at least 100,000 image samples for training the model. It is also important to pick the dataset with good accurate annotations. So, we have chosen text recognition data from AcademicTorrents.

Some of the important modules required for this project are:

1.Tensorflow

2.Keras

3.Numpy

4.Open

5.Matplotlib Etc.

Unlike the classically structured datasets, we have an image dataset as input. Hence we won't be having numeric or text-related columns. But any model cannot process an image directly to compute the predictions. We first need to perform the featurization. Featurization is nothing but converting the image into an n-dimensional vector and we pass that n-dimensional vector to the neural network in order to get the predictions. Typically we convert the image into a NumPy array.

The technology offers a comprehensive solution for document capture and form processing. OCR typically employs a modular, open, scalable, and workflow-controlled design. It has capabilities for defining shapes, scanning, picture pre-processing, and recognition.

* The following is a summary of the main characteristics of OCR scanners based on machine learning.

* The item will assist the user in converting handwritten text into machine-encoded text.

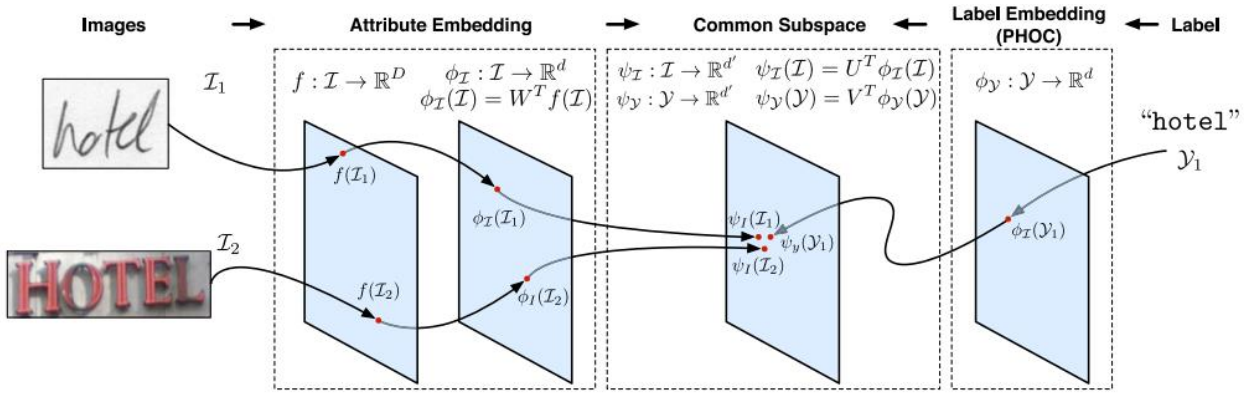
* This will make it easier for the user to keep their content in an editable state without worrying about maintaining it.

* It will be simpler to search for any text or phrase since users may turn their data into machine-encoded text.

* Data loss or unauthorized access to documents in paper format may be avoided with the help of this method.

* In paper format documents, once they are destroyed, data recovery is not feasible. However, if they are transformed into computer text documents, recovery is still possible with the right back-up procedures in place.

Project Workflow :



3.Related Work (Background)

When it comes to optical character recognition (OCR), there are mainly two kinds currently available.

A)There are some freely available python packages by using which we can perform OCR, but more often we will get less accuracy. Some example packages are 1) Pytesseract and 2) Paddleocr Etc.,

B)We need to train deep learning models from scratch. For optical character recognization, the most popular deep learning architecture is a Convolution recurrent neural network.

4.Dataset

In order to find a relevant good dataset we did a lot of research. Most of the available datasets on the internet are having a lot of noisy data and blurry images. Finally, we found a good dataset. It has a huge number of gray-scale word-level images. The link to the dataset is available below.

https://www.robots.ox.ac.uk/~vgg/data/text/#sec-synth

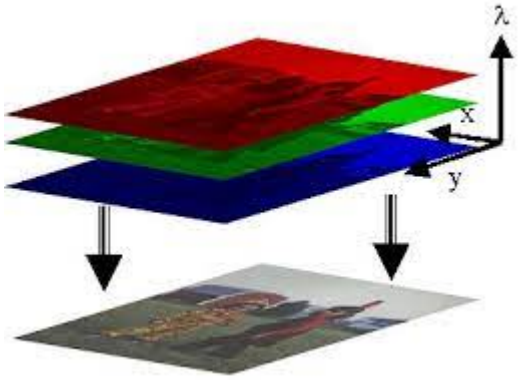
If you visit the above link you can find download the synthetic English word dataset. In the dataset, they have more than 9 Million images and 90k English words. This data is very huge and it consumes a lot of

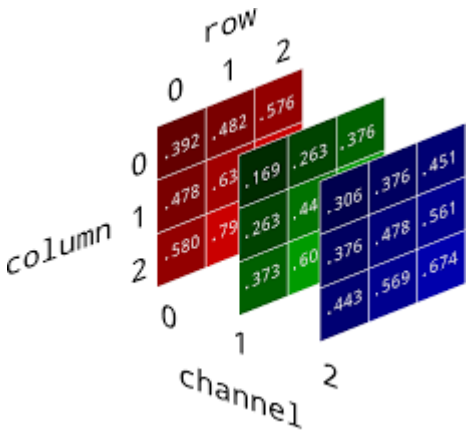
time and space complexity. Hence we downloaded the whole dataset and filtered around 150,000 images. We are using the same dataset for both training and testing.

5.Detail design of Features

If we have a text dataset they are a lot of featurization techniques are available. Since we are dealing with the image dataset here we have to convert the given image into a multi-dimensional NumPy array. Generally, a colored image contains 3 layers of RGB with shape (height, width,3). But in our training dataset, we have a grayscale image. Hence the dataset shape will be (height, width)

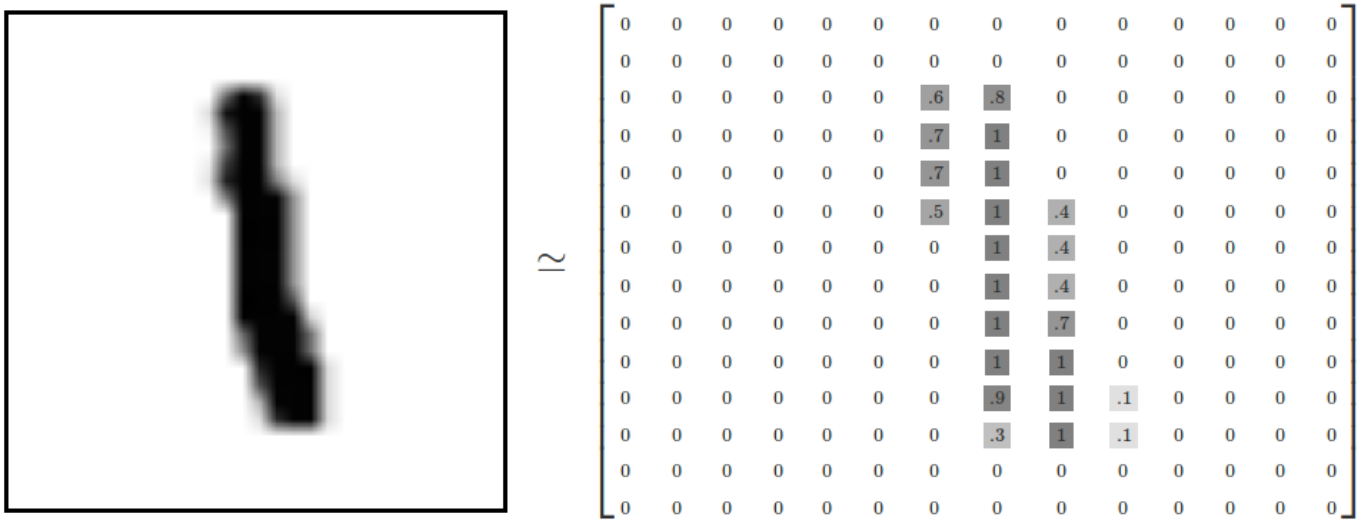
RGB Image Representation in Numpy array





Gray Scale to array representation:

The below image represents the featurization of the Gray Scale image to Numpy array.

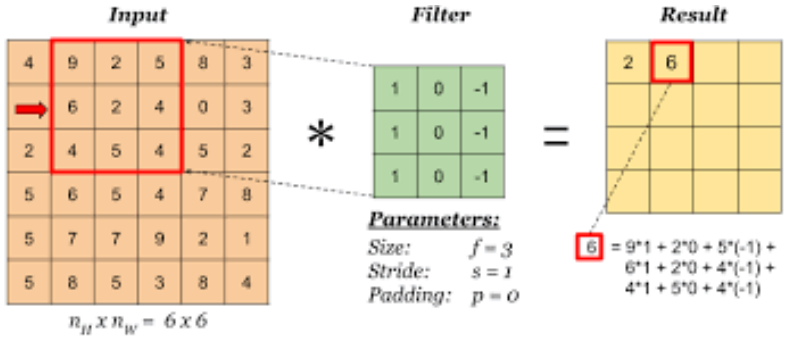


6.Analysis

As we already mentioned we are using a convolution recurrent neural network. It contains mainly two parts 1) Convolution and 2) Recurrent neural network

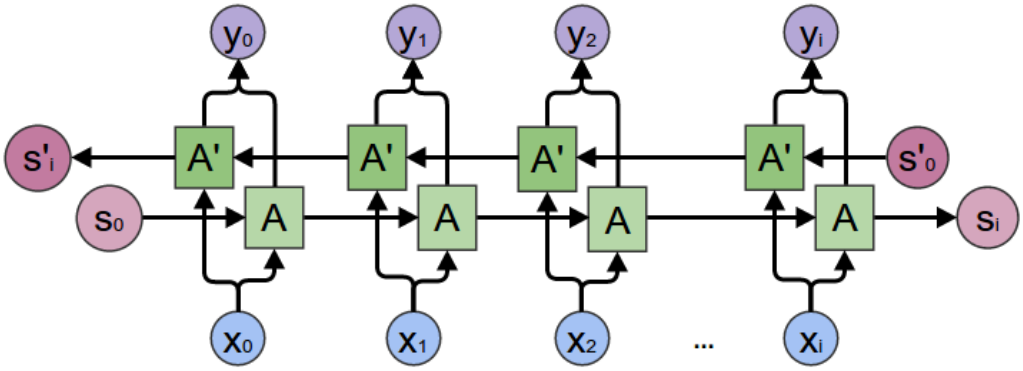
1)Convolution:

Convolution is an operation that mainly helps to understand the shapes in the given image. In simple words, it will remove the unnecessary objects in the image and focuses more on the character/ Object in the image.



2)Recurrent neural network:

It is a very popular and widely used network when it comes to dealing with NLP tasks. Recurrent neural got popular when the research paper called Attention is published. The recurrent neural network basically focuses on both previous and upcoming characters (Bi-directional LSTM). The below screenshot is the architecture of Bi-directional LSTM.



7.Implementation

Here in this section, we have mainly 3 Modules.

1)Loading dataset into Google Colab.

As we already discussed we have filtered approximately 150,000 images and created a Zip File for those images. The best-suggested way is to upload the corresponding Zip file into google drive and we can access to the Zip file by mounting the google drive using google colab. It is easy to unzip a Zip file it just requires a simple Linux Command

```
!unzip /content/drive/MyDrive/90kDICT32px.zip -d /content/90kDICT32px/
  inflating: /content/90kDICT32px/9/6/190_DOMINO_23217.jpg
  inflating: /content/90kDICT32px/9/6/191_GROUNDSHEET_33910.jpg
  inflating: /content/90kDICT32px/9/6/192_ORNAMENTS_53783.jpg
  inflating: /content/90kDICT32px/9/6/193_MAYWEATHER_47300.jpg
  inflating: /content/90kDICT32px/9/6/194_Wrangled_87352.jpg
  inflating: /content/90kDICT32px/9/6/195_Unabashed_81674.jpg
```

2)Data Preprocessing:

In this section, some necessary prepossessing steps are required to get the important variables like the label, array representation of the image, input length, padding of the label, etc. If you observe the above screenshot, the label name is present in the file name itself. Hence we can extract the label name using "filename.split("_")[1]". This will return the label for the corresponding image.

3)Model Building:

As discussed earlier, we are using the Convolution Recurrent Neural Network for our model building. We have tried several combinations of layers of both convolution and Bi-directional LSTM layers. We have chosen a particular model which gives us minimal loss. In our model, we used the 7 convolution layers and 2 bi-directional LSTM layers. The below screenshot represents the model that gives best result.

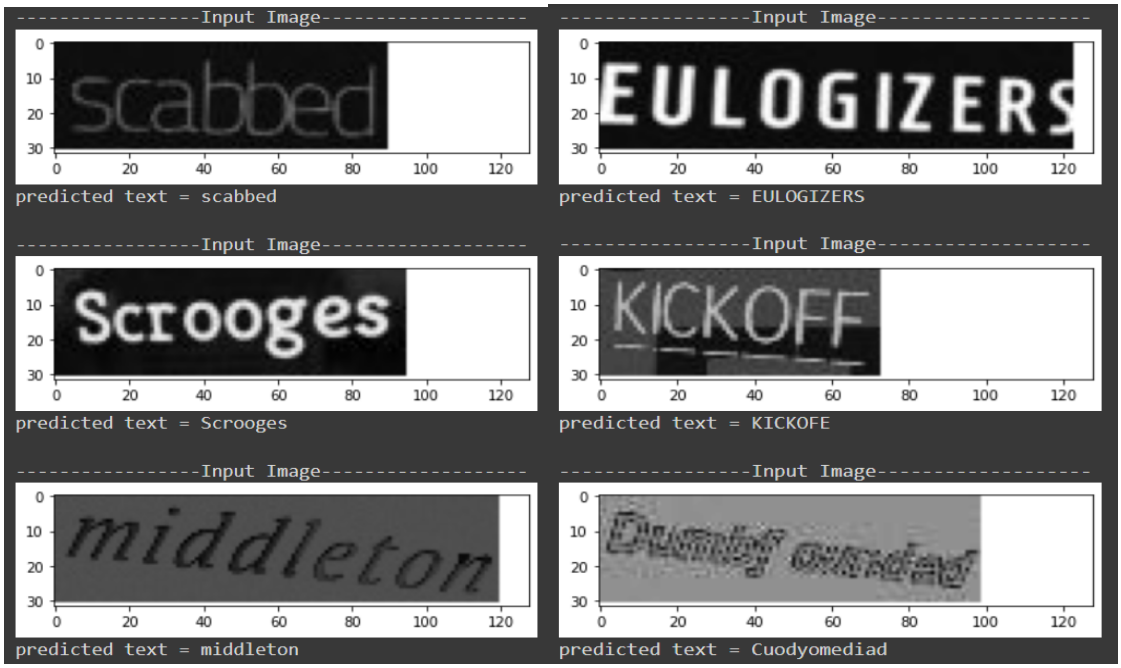
Model Architecture

```
inputs = Input(shape=(32,128,1))
conv_1 = Conv2D(64, (3,3), activation = 'relu', padding='same')(inputs)
pool_1 = MaxPool2D(pool_size=(2, 2), strides=2)(conv_1)
conv_2 = Conv2D(128, (3,3), activation = 'relu', padding='same')(pool_1)
pool_2 = MaxPool2D(pool_size=(2, 2), strides=2)(conv_2)
conv_3 = Conv2D(256, (3,3), activation = 'relu', padding='same')(pool_2)
conv_4 = Conv2D(256, (3,3), activation = 'relu', padding='same')(conv_3)
pool_4 = MaxPool2D(pool_size=(2, 1))(conv_4)
conv_5 = Conv2D(512, (3,3), activation = 'relu', padding='same')(pool_4)
batch_norm_5 = BatchNormalization()(conv_5)
conv_6 = Conv2D(512, (3,3), activation = 'relu', padding='same')(batch_norm_5)
batch_norm_6 = BatchNormalization()(conv_6)
pool_6 = MaxPool2D(pool_size=(2, 1))(batch_norm_6)
conv_7 = Conv2D(512, (2,2), activation = 'relu')(pool_6)
squeezed = Lambda(lambda x: K.squeeze(x, 1))(conv_7)
blstm_1 = Bidirectional(LSTM(128, return_sequences=True, dropout = 0.2))(squeezed)
blstm_2 = Bidirectional(LSTM(128, return_sequences=True, dropout = 0.2))(blstm_1)
outputs = Dense(len(characterset)+1, activation = 'softmax')(blstm_2)

actual_model = Model(inputs, outputs)
```

8.Preliminary Results

After training the model, we have done predictions of some sample images. The below screen-shots show some of the predicted outputs.



9.Project Management

Work Completed:

Everyone has contributed equally to our project.

- All the datasets required for the project are analyzed and created by Neha Kalluri.
- Converting images to a multidimensional array is done by Lahari Kunduru. This is done to train the machine learning model based on the features of the image.
- Data Preprocessing is done by Keerthana Pinikeshi which includes an array representation of the image and padding of the label.
- Convolution Recurrent Neural Network implementation for model building is taken care of by Neha.
- The documentation part is done by Keerthana Pinikeshi and Lahari Kunduru.

What to be completed:

- In Data preprocessing, we have to still remove slants as slants vary from user to user and we have to convert them to standard. Keerthana will be working on that.
- We have to also identify connected components and identify them as individual characters. It will be taken care of by Neha.
- We have to check for new features to improve the performance of the current model which will be done by Lahari.

Concerns:

- If we see the output for the sample inputs, we have to still update our code to recognize the letters.
- Sometimes, F is also recognized as E, and C is considered as E. We have to still work on it for better results.

10.References:

Below are some of the initial references that we have researched as part of this project.

https://en.wikipedia.org/wiki/Optical_character_recognition

https://www.flatworldsolutions.com/data-management/articles/key-advantages-ocr-based-data-entry.php

https://www.cloudfactory.com/machine-learning/optical-character-recognition-ocr

https://www.linkedin.com/pulse/15-best-ocr-handwriting-datasets-machine-learning-limarc-ambalina/