Project Report

Telugu OCR

Srinivasula Koushik - 18277

Desaraju Harsha Vardhan - 18084

14th December 2020

Idea

In recent times, we have seen that OCR(Optical Character Recognition) is already quite developed for English and our Indian languages fall behind in this regard. So we chose Telugu for this project.

OCR is identifying the text in an image.

In this project we take an image which consists of Telugu text and extract the text from it.

But it is not as easy as English as the script for Telugu is very different from English. In general Indian languages are agglutinative in nature. So it becomes more difficult as we have to look from more than an alphabet to identify a character.

Steps Involved:-

A) Preprocessing
   a) Skew correction of the image
   b) Binarization
   c) Line level segmentation
   d) Word level segmentation
   e) Character level segmentation

B) Character Identification
   a) Model

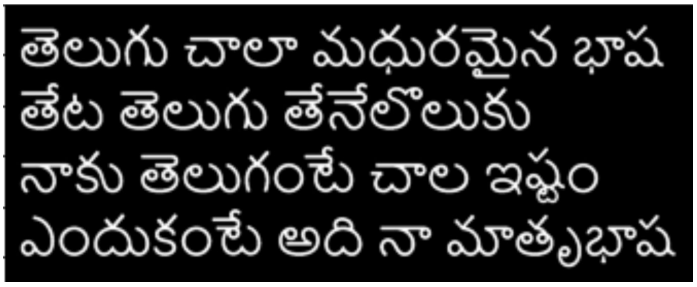
C) Output/Evaluation

Preprocessing:-

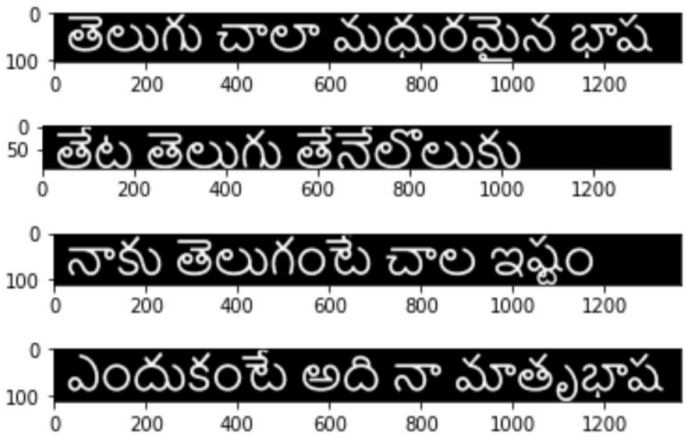
Preprocessing is converting the image containing the text to a set of images containing characters for the model to identify. It contains 5 steps.

a) **Skew correction:** Text in the image can be tilted with respect to the image, which has to be corrected for better performance of the model.
b) **Binarization:** It is the process of converting the pixel values of the image to 0 or 1.

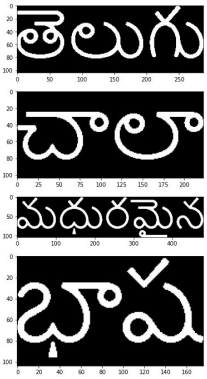
Image after Skew correction and binarization.



c) **Line Level Segmentation:** It is splitting the image into a set of lines.But this is not as trivial as in English. In Telugu there are ‘Vattu’ and ‘Gunintam’ for adding consonants and vowel sounds respectively. In many cases ‘vattu’ is not directly connected to the main character which can be problematic. In lines containing ‘vattu’, the line and vattu alone can be divided into different lines. So a different approach has to be used than for English.



d) **Word level Segmentation:** It is dividing a line into different words.



e) **Character level Segmentation:** It is dividing a word into its constituent letters.

The approach we used for line level segmentation is to calculate a minimum distance above which the image is split. This is calculated on the go from the image. This is based on the fact that the distance between character and vattu is less than distance between two lines.

Model:

In Telugu language, As we have seen in preprocessing. We need to train two models.

1) For main character
2) For Vattu and Gunintham

The models are trained on a dataset of nearly 67 Lakh samples of Telugu characters.

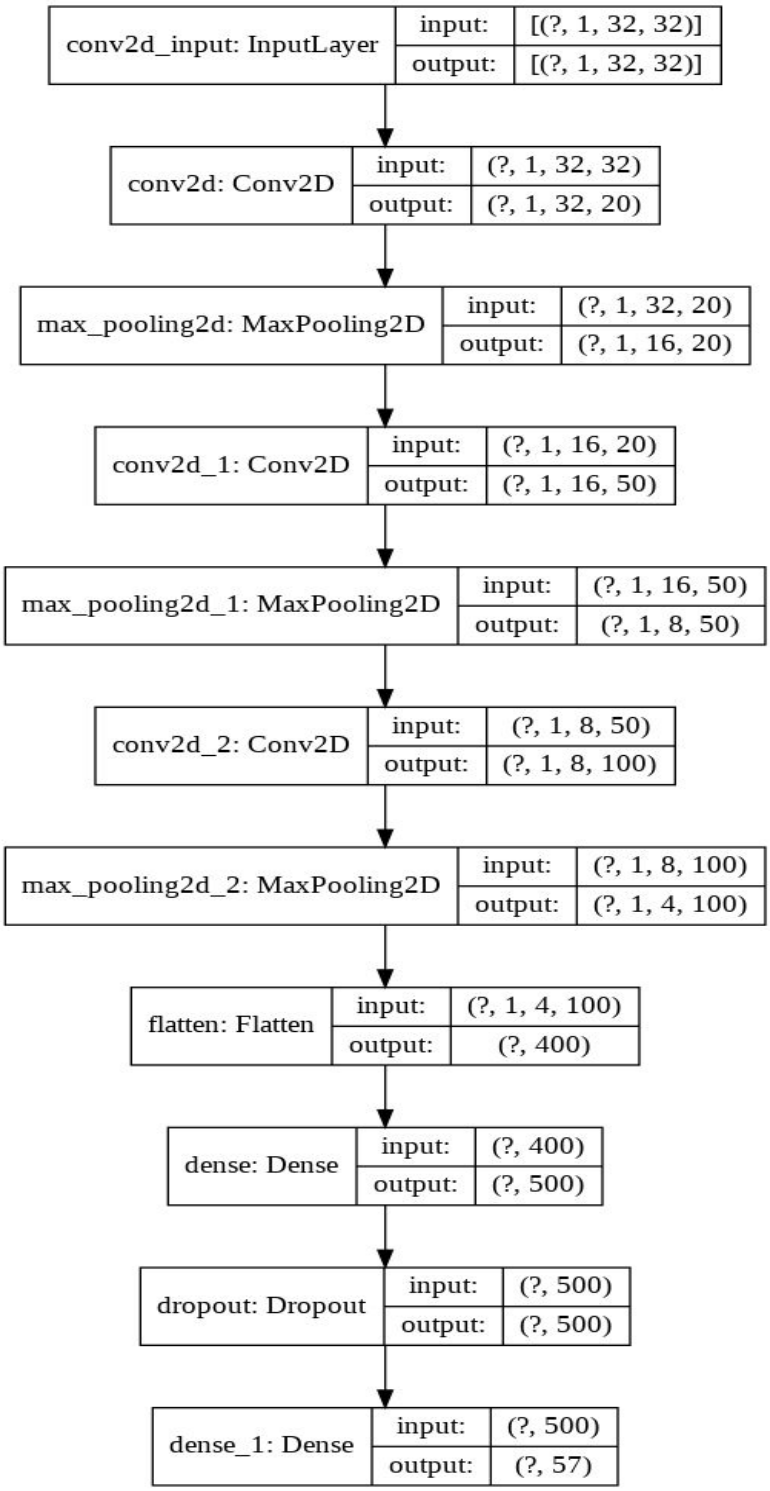
Each image is a numpy array of size 32x32 of telugu characters.

In the Main character, Labels are 1-57 representing the unique main characters. And in Vattu and gunintham Labels are 1-541 representing the unique vattu and gunintham combinations.

Architecture of CNNs:

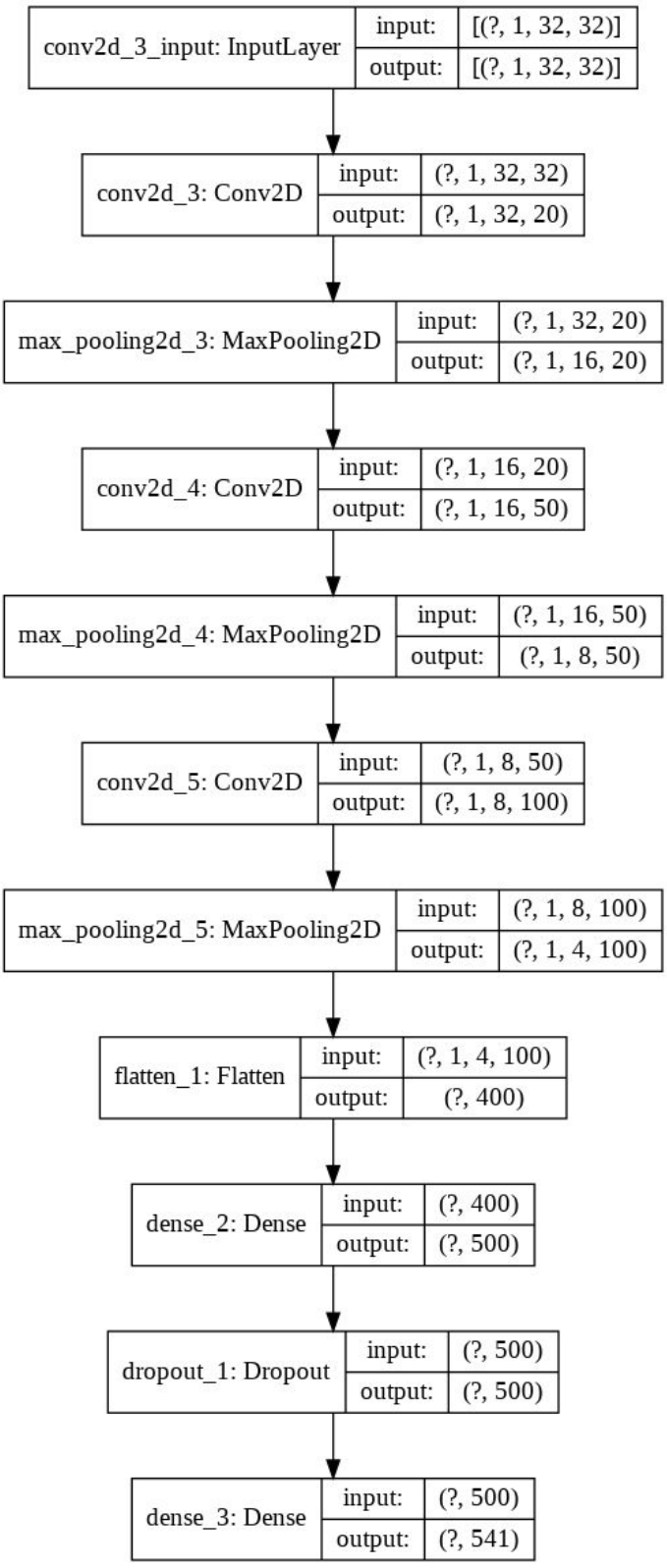
MAIN CHAR MODEL:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 1, 32, 20)         5780
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 1, 16, 20)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 1, 16, 50)         9050
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 1, 8, 50)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 1, 8, 100)         45100
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 4, 100)         0
_________________________________________________________________
flatten (Flatten)            (None, 400)               0
_________________________________________________________________
dense (Dense)                (None, 500)               200500
_________________________________________________________________
dropout (Dropout)            (None, 500)               0
_________________________________________________________________
dense_1 (Dense)              (None, 57)                28557
=================================================================
Total params: 288,987
Trainable params: 288,987
Non-trainable params: 0
```


conv2d_input: InputLayer
input: [(?, 1, 32, 32)]
output: [(?, 1, 32, 32)]
conv2d: Conv2D
input: (?, 1, 32, 32)
output: (?, 1, 32, 20)
max_pooling2d: MaxPooling2D
input: (?, 1, 32, 20)
output: (?, 1, 16, 20)
conv2d_1: Conv2D
input: (?, 1, 16, 20)
output: (?, 1, 16, 50)
max_pooling2d_1: MaxPooling2D
input: (?, 1, 16, 50)
output: (?, 1, 8, 50)
conv2d_2: Conv2D
input: (?, 1, 8, 50)
output: (?, 1, 8, 100)
max_pooling2d_2: MaxPooling2D
input: (?, 1, 8, 100)
output: (?, 1, 4, 100)
flatten: Flatten
input: (?, 1, 4, 100)
output: (?, 400)
dense: Dense
input: (?, 400)
output: (?, 500)
dropout: Dropout
input: (?, 500)
output: (?, 500)
dense_1: Dense
input: (?, 500)
output: (?, 57)

VATTU and GUNINTAM:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 1, 32, 20)         5780
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 16, 20)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 1, 16, 50)         9050
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 1, 8, 50)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 1, 8, 100)         45100
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 4, 100)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_2 (Dense)              (None, 500)               200500
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_3 (Dense)              (None, 541)               271041
=================================================================
Total params: 531,471
Trainable params: 531,471
Non-trainable params: 0
```


conv2d_3_input: InputLayer
input: [(?, 1, 32, 32)]
output: [(?, 1, 32, 32)]
conv2d_3: Conv2D
input: (?, 1, 32, 32)
output: (?, 1, 32, 20)
max_pooling2d_3: MaxPooling2D
input: (?, 1, 32, 20)
output: (?, 1, 16, 20)
conv2d_4: Conv2D
input: (?, 1, 16, 20)
output: (?, 1, 16, 50)
max_pooling2d_4: MaxPooling2D
input: (?, 1, 16, 50)
output: (?, 1, 8, 50)
conv2d_5: Conv2D
input: (?, 1, 8, 50)
output: (?, 1, 8, 100)
max_pooling2d_5: MaxPooling2D
input: (?, 1, 8, 100)
output: (?, 1, 4, 100)
flatten_1: Flatten
input: (?, 1, 4, 100)
output: (?, 400)
dense_2: Dense
input: (?, 400)
output: (?, 500)
dropout_1: Dropout
input: (?, 500)
output: (?, 500)
dense_3: Dense
input: (?, 500)
output: (?, 541)

Model accuracy:

1) Main Character: 97.7
2) Vattu_gunintham: 95.8

These models are now used to predict the telugu text from the image.

OUTPUT:

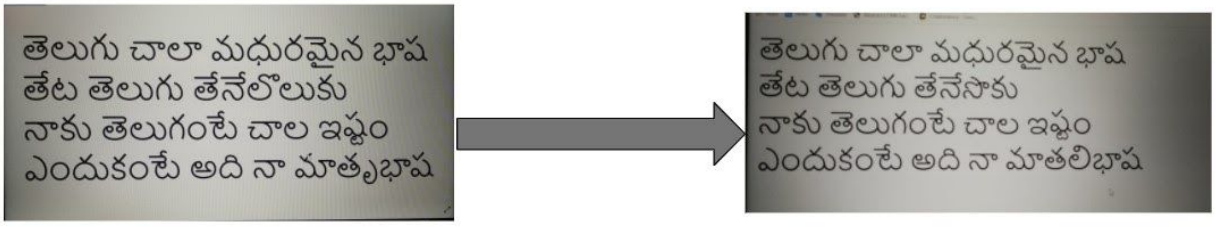
Our input is an image consisting of telugu text .

We first preprocess it as explained above and then feed it to the model for prediction.

Model returns the unique tags for both main and vattu_gunintham characters and we then match those tags with their respective characters and reconstruct the full text.

Here, we printed the output in an HTML file as printing it in python or text file is not supported for Telugu without special packages.

Here is a demonstration for the output from an input.



INPUT: Image format

OUTPUT: Text format in HTML

తెలుగు చాలా మధురమైన భాష
తేట తెలుగు తేనేలొలుకు
నాకు తెలుగంటే చాల ఇష్టం
ఎందుకంటే అది నా మాతృభాష

తెలుగు చాలా మధురమైన భాష
తేట తెలుగు తేనేసొకు
నాకు తెలుగంటే చాల ఇష్టం
ఎందుకంటే అది నా మాతలిభాష

Compare the underlined text in both the input and output image. The difference is because, in the second line there is no gap between the second letter and the third letter and both of them are treated as one character and the prediction is made of that image. See fig 1 below.

Figure 1

Figure 2

In the last line, the letter which is boxed in the above image (fig 2), should be treated as one character but as there is a gap between the main character and the ‘gunintam’, they are treated as separate letters and predictions are made for both the images.

References:-

1) Paper:- https://arxiv.org/abs/1711.07245
2) Dataset: The same as the paper has used