DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SREEPATHY INSTITUTE OF MANAGEMENT & TECHNOLOGY VAVANOOR
PALAKKAD PIN : 679533
CS492 - Project External Review

Guide: Sreeshma.K
Date : 28.06.2022

Group No: 07
Aswathy.S [TL]
Aswin.V
Visakh.M
Muhammed Nishad.K.P

Malayalam Handwritten Text Recognition using Machine Learning

Introduction.

- This project deals with the goal of recognizing handwritten malayalam text and output them as digitally processable text format.
- Handwriting recognition technologies has advanced in languages like English, Japanese, Mandarin but Malayalam is still a long way to go.
- Existing recognition models Malayalam is not very accurate and most requires printed Malayalam texts to work.
- If a reliable model can be developed for recognizing Malayalam handwritten texts, it can be used to convert many ancient documents, legal documents, and several other manuscripts that are physically stored and are in poor state.

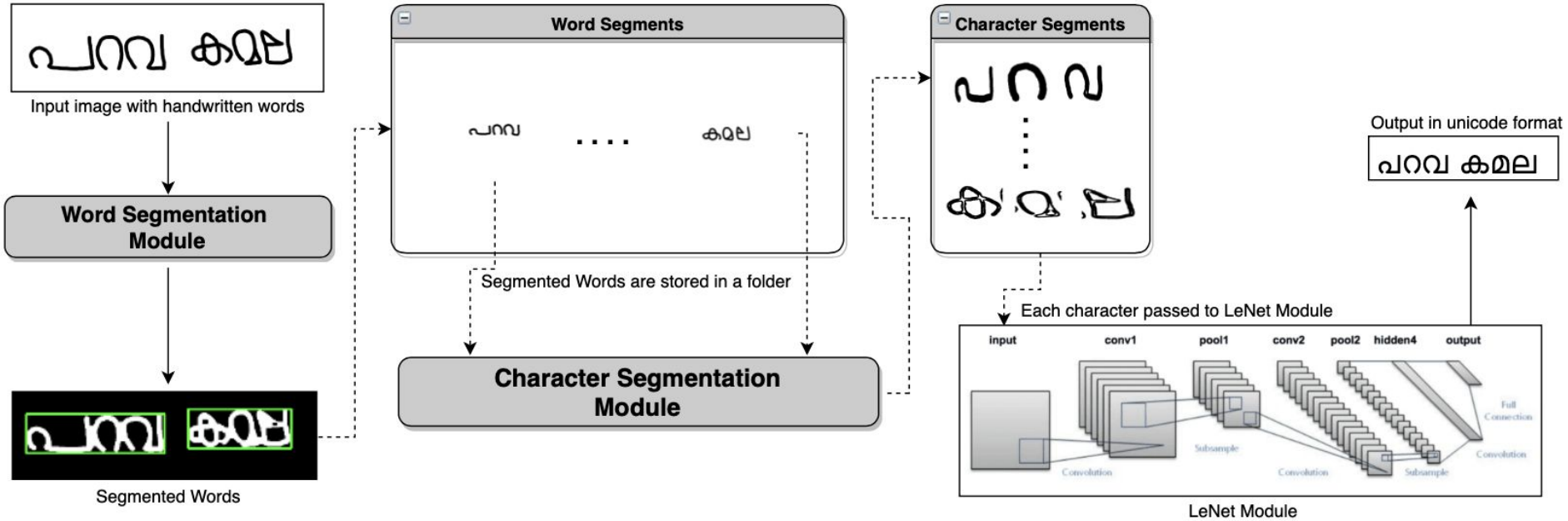
Introduction.

- Applications of handwriting recognition are numerous: reading postal addresses, bank check amounts, and forms.
- Furthermore, OCR plays an important role for digital libraries, allowing the entry of image textual information into computers by digitization, image restoration, and recognition methods.
- The first challenge is to segment the words.
- The second challenge is to segment that word into individual characters.
- The third challenge is to build a model capable of recognizing and generalizing diverse handwriting styles.

Objectives.

- Develop a benchmarking dataset of Handwritten Malayalam words.
- The main objective of this research is to develop recognition methods for handwritten Malayalam words.
- Develop methods for script independent handwritten recognition.
- To identify the best Deep Learning approach for the recognition of handwritten documents.

Architectural Diagram



Architectural Diagram of MHR

Word Segmentation

- Word segmentation is the process of detecting and extracting individual words from a line of text given as the input.

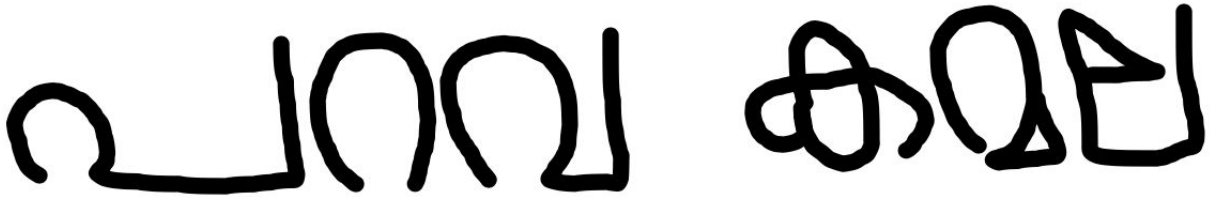


Fig: Input Image

Fig: Bounding box during Segmentation

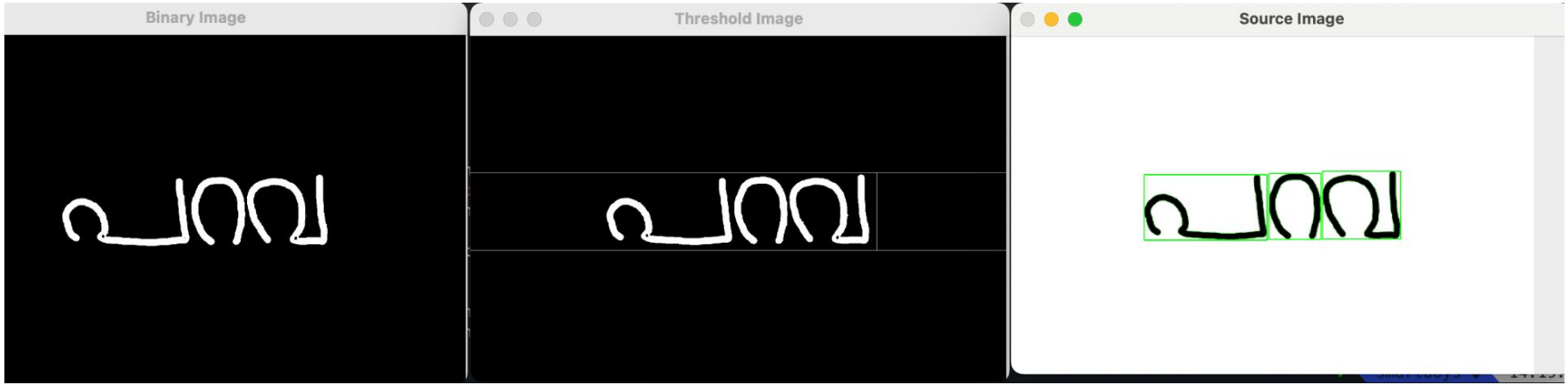
Fig: Result after Word Segmentaion

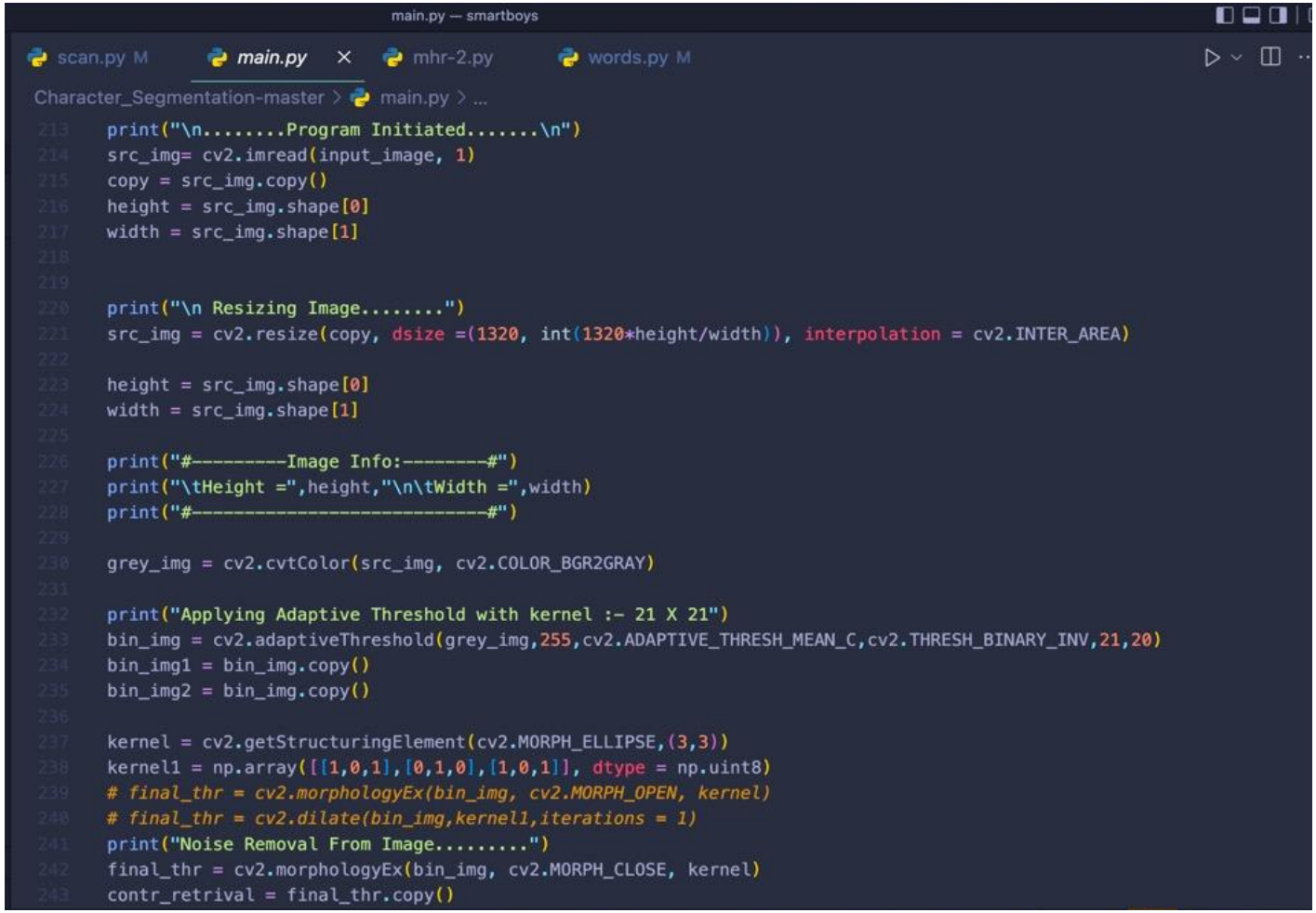
word_segmentation > words.py > ...

```python
def _text_detect(img, image, join=False):
    """Text detection using contours."""
    small = resize(img, 2000)

    # Finding contours
    # mask = np.zeros(small.shape, np.uint8)
    kernel = np.ones((5, 100), np.uint16)  ### (5, 100) for line segmention  (5,30) for word segmentation
    img_dilation = cv2.dilate(small, kernel, iterations=1)
    # print(11111111111111)

    # im2,
    cnt, hierarchy = cv2.findContours(np.copy(small),
                                      cv2.RETR_TREE,
                                      cv2.CHAIN_APPROX_SIMPLE)

    index = 0
    boxes = []
    # Go through all contours in top level
    while (index >= 0):
        x,y,w,h = cv2.boundingRect(cnt[index])
        cv2.drawContours(img_dilation, cnt, index, (255, 255, 255), cv2.FILLED)
        maskROI = img_dilation[y:y+h, x:x+w]
        # Ratio of white■ pixels to area of bounding rectangle
        r = cv2.countNonZero(maskROI) / (w * h)

        # Limits for text
        if (r > 0.1
            and 1600 > w > 10
            and 1600 > h > 10
            and h/w < 3
```

Character Segmentation

- Character segmentation refers to the process of dividing a word or a line into individual characters.
- Unlike line or word segmentation, the same approach can't always be used for character segmentation of several languages because different languages use different types of characters.




main.py — smartboys
scan.py M
main.py
mhr-2.py
words.py M
Character_Segmentation-master > main.py > ...
print("\n........Program Initiated.......\n")
src_img= cv2.imread(input_image, 1)
copy = src_img.copy()
height = src_img.shape[0]
width = src_img.shape[1]
print("\n Resizing Image.........")
src_img = cv2.resize(copy, dsize =(1320, int(1320*height/width)), interpolation = cv2.INTER_AREA)
height = src_img.shape[0]
width = src_img.shape[1]
print("#--------Image Info:--------#")
print("\tHeight =",height,"\n\tWidth =",width)
print("#---------------------------#")
grey_img = cv2.cvtColor(src_img, cv2.COLOR_BGR2GRAY)
print("Applying Adaptive Threshold with kernel :- 21 X 21")
bin_img = cv2.adaptiveThreshold(grey_img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY_INV,21,20)
bin_img1 = bin_img.copy()
bin_img2 = bin_img.copy()
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
kernel1 = np.array([[1,0,1],[0,1,0],[1,0,1]], dtype = np.uint8)
# final_thr = cv2.morphologyEx(bin_img, cv2.MORPH_OPEN, kernel)
# final_thr = cv2.dilate(bin_img,kernel1,iterations = 1)
print("Noise Removal From Image.........")
final_thr = cv2.morphologyEx(bin_img, cv2.MORPH_CLOSE, kernel)
contr_retrival = final_thr.copy()

Character Recognition

Lenet-5 Architecture Implementation

Layers

- Convolution, Activation, and pooling
- Convolution, Activation, and pooling
- Fully-connected
- Activation
- Fully-connected
- Softmax classifier

Lenet-5 Architecture Implementation

- The LeNet-5 architecture consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a softmax classifier.
- The input for LeNet-5 is a 32×32 grayscale image which passes through the first convolutional layer with 6 feature maps or filters having size 5×5 and a stride of one. The image dimensions changes from 32x32x1 to 28x28x6.
- Then the LeNet-5 applies average pooling layer or sub-sampling layer with a filter size 2×2 and a stride of two. The resulting image dimensions will be reduced to 14x14x6.
- Next, there is a second convolutional layer with 16 feature maps having size 5×5 and a stride of 1.

Lenet-5 Architecture Implementation

- The fourth layer (S4) is again an average pooling layer with filter size 2×2 and a stride of 2. This layer is the same as the second layer (S2) except it has 16 feature maps so the output will be reduced to 5x5x16.
- The fifth layer (C5) is a fully connected convolutional layer with 120 feature maps each of size 1×1. Each of the 120 units in C5 is connected to all the 400 nodes (5x5x16) in the fourth layer S4.
- The sixth layer is a fully connected layer (F6) with 84 units.
- Finally, there is a fully connected softmax output layer with possible values of characters.

Neural Network

- Convolution
  - Input : 1x1x32x32
  - Features : 32
  - Filter : 3x3
  - Output : 1x32x30x30
- Convolution
  - Input : 1x32x30x30
  - Features : 64
  - Filter : 3x3
  - Output : 1x64x28x28
- Max-Pooling
  - Input : 1x64x28x28
  - Kernel : 2x2
  - Output : 1x64x14x14

Neural Network

- Convolution
  - Input : 1x64x14x14
  - Features : 128
  - Filter : 3x3
  - Output : 1x128x12x12
- Max-Pooling
  - Input : 1x128x12x12
  - Kernel : 2x2
  - Output : 1x128x6x6
- Fully-Connected
  - Input : 1x128x6x6
  - Output : 1x4608

Neural Network

- Dense
  - Input : 1x4608
  - Classes : 128
  - Activation = relu
  - Output : 1x128
- Dense
  - Input : 1x128
  - Classes : 48
  - Activation = softmax
  - Output : 1x48

Training the Model

Dataset

- We have collected different handwritten samples of malayalam alphabets in separate folders for each characters.

Data Processing

- Sorting images by characters
- Cleaning images
- 100 - 150 images per character
- Augmenting to 500 images per character
- Converting to 32x32

Libraries Used.

- Keras
- OpenCV
- Tensorflow
- NumPy

Augmenting

- Augmenting Images

```
folder = 'raw_data'
for f in list_folders(folder):
    if os.path.isdir(os.path.join(folder, f, 'output')):
        shutil.rmtree(os.path.join(folder, f, 'output'))

    # creating a pipeline object to apply operations on it
    p = Augmentor.Pipeline(os.path.join(folder, f))
    p.random_distortion(probability=1, grid_width=10, grid_height=10, magnitude=8)
    p.sample(500, multi_threaded=False)
```

- Two operations are performed
    - Random Distortion
    - Sample

Augmenting

- Random Distortion
  - This function performs a randomised elastic distortion.
  - The grid width and height controls how fine the distortions are.
  - The magnitude of the distortions can be controlled using magnitude.
- Sample
  - This function samples from the pipeline, using the original images defined during instantiation.
  - All images generated by the pipeline are by default stored in an output directory, relative to the path defined during the pipeline's instantiation.

Data Augment

```
Augment the data : `python dataset_augmentor.py`
Crop and resize the images : `python data_cleaner.py`
Process the images : `python data_process.py`
Train the network : `python train.py`
Predict an image : `python scan.py -i <filename>`
vishakh@Vishakhs-MacBook-Pro malayalam-character-recognition % python3 dataset_augmentor.py
Initialised with 198 image(s) found.
Processing 218.png: 100%|| 500/500 [00:11<00:00, 43.46 Samples/s]
Initialised with 196 image(s) found.
Processing 26.png: 100%|| 500/500 [00:11<00:00, 42.99 Samples/s]
Initialised with 198 image(s) found.
Processing 104.png: 100%|| 500/500 [00:11<00:00, 44.73 Samples/s]
Initialised with 182 image(s) found.
Processing 70.png: 100%|| 500/500 [00:10<00:00, 45.96 Samples/s]
Initialised with 179 image(s) found.
Processing 20.png: 100%|| 500/500 [00:11<00:00, 45.29 Samples/s]
Initialised with 177 image(s) found.
Processing 7.png:  68%|| 338/500 [00:07<00:03, 44.04 Samples/s
```

Data Cleaning - Cropping and Resizing the Images

```
def list_folders(root_folder):···

def create_folders(root_folder, folder_list):···

def read_transparent_png(filename):···

def clean(img):···

def crop(image, desired_size):···

def process_folder(folder):···

def save_new(folder, imglist):···

def process_images(raw_folder, clean_folder, folder_list):···

def skeletize(img):···
```

Data Cleaning

```
vishakh@Vishakhs-MacBook-Pro malayalam-character-recognition % python3 data_cleaner.py
['character_3333', 'character_3334', 'character_3335', 'character_3337', 'character_3342', 'character_3343', 'character_3346', 'character_3349', 'character_3350', 'character_3351', 'character_3352', 'character_3353', 'character_3354', 'character_3355', 'character_3356', 'character_3357', 'character_3358', 'character_3359', 'character_3360', 'character_3361', 'character_3362', 'character_3363', 'character_3364', 'character_3365', 'character_3366', 'character_3367', 'character_3368', 'character_3370', 'character_3371', 'character_3372', 'character_3373', 'character_3374', 'character_3375', 'character_3376', 'character_3377', 'character_3378', 'character_3379', 'character_3380', 'character_3381', 'character_3382', 'character_3383', 'character_3384', 'character_3385', 'character_3450', 'character_3451', 'character_3452', 'character_3453', 'character_3454']
character_3333
character_3334
character_3335
character_3337
character_3342
character_3343
character_3346
character_3349
character_3350
character_3351
character_3352
character_3353
character_3354
character_3355
character_3356
character_3357
character_3358
character_3359
character_3360
character_3361
character_3362
character_3363
character_3364
character_3365
character_3366
character_3367
character_3368
character_3370
character_3371
character_3372
character_3373
character_3374
character_3375
character_3376
character_3377
character_3378
character_3379
character_3380
character_3381
character_3382
character_3383
character_3384
character_3385
character_3450
character_3451
character_3452
character_3453
character_3454
vishakh@Vishakhs-MacBook-Pro malayalam-character-recognition %
```

Data Process - Process the images

```
vishakh@Vishakhs-MacBook-Pro malayalam-character-recognition % python3 data_process.py
data/character_3333
Full dataset tensor: (1000, 32, 32)
Mean: 0.123986326
Standard deviation: 0.32956597
data/character_3334
Full dataset tensor: (1000, 32, 32)
Mean: 0.12946582
Standard deviation: 0.3357148
data/character_3335
Full dataset tensor: (1000, 32, 32)
Mean: 0.14703418
Standard deviation: 0.35413998
data/character_3337
Full dataset tensor: (1000, 32, 32)
Mean: 0.12996778
Standard deviation: 0.336268
data/character_3342
Full dataset tensor: (1000, 32, 32)
Mean: 0.1169873
Standard deviation: 0.32140517
data/character_3343
Full dataset tensor: (1000, 32, 32)
Mean: 0.11399805
Standard deviation: 0.31780887
data/character_3346
Full dataset tensor: (1000, 32, 32)
Mean: 0.119210936
Standard deviation: 0.32403654
data/character_3349
Full dataset tensor: (1000, 32, 32)
Mean: 0.13465919
Standard deviation: 0.34135923
data/character_3350
Full dataset tensor: (1000, 32, 32)
Mean: 0.12053223
Standard deviation: 0.32558286
data/character_3351
Full dataset tensor: (1000, 32, 32)
Mean: 0.0971582
Standard deviation: 0.29617304
data/character_3352
Full dataset tensor: (1000, 32, 32)
Mean: 0.106933594
Standard deviation: 0.30902874
data/character_3353
Full dataset tensor: (1000, 32, 32)
Mean: 0.11867578
Standard deviation: 0.3234066
data/character_3354
Full dataset tensor: (1000, 32, 32)
Mean: 0.09006738
Standard deviation: 0.28627837
data/character_3355
Full dataset tensor: (1000, 32, 32)
Mean: 0.09913086
Standard deviation: 0.2988376
data/character_3356
Full dataset tensor: (1000, 32, 32)
Mean: 0.16902246
Standard deviation: 0.37477177
data/character_3357
Full dataset tensor: (1000, 32, 32)
```

```
Standard deviation: 0.28257918
data/character_3378
Full dataset tensor: (1000, 32, 32)
Mean: 0.124131836
Standard deviation: 0.32973188
data/character_3379
Full dataset tensor: (1000, 32, 32)
Mean: 0.1401504
Standard deviation: 0.34714296
data/character_3380
Full dataset tensor: (1000, 32, 32)
Mean: 0.11162793
Standard deviation: 0.31490812
data/character_3381
Full dataset tensor: (1000, 32, 32)
Mean: 0.1078252
Standard deviation: 0.3101595
data/character_3382
Full dataset tensor: (1000, 32, 32)
Mean: 0.11478125
Standard deviation: 0.31875777
data/character_3383
Full dataset tensor: (1000, 32, 32)
Mean: 0.124625
Standard deviation: 0.33029324
data/character_3384
Full dataset tensor: (1000, 32, 32)
Mean: 0.10010745
Standard deviation: 0.3000143
data/character_3385
Full dataset tensor: (1000, 32, 32)
Mean: 0.08516211
Standard deviation: 0.27912283
data/character_3450
Full dataset tensor: (1000, 32, 32)
Mean: 0.11314258
Standard deviation: 0.316767
data/character_3451
Full dataset tensor: (1000, 32, 32)
Mean: 0.11541699
Standard deviation: 0.31952447
data/character_3452
Full dataset tensor: (1000, 32, 32)
Mean: 0.104831055
Standard deviation: 0.30633563
data/character_3453
Full dataset tensor: (1000, 32, 32)
Mean: 0.13210547
Standard deviation: 0.33860534
data/character_3454
Full dataset tensor: (1000, 32, 32)
Mean: 0.11962988
Standard deviation: 0.32452828
48
50 100 350
0 50
50 150
150 500
Training set (16800, 32, 32) (16800,)
Test set (4800, 32, 32) (4800,)
Validation set (2400, 32, 32) (2400,)
Compressed pickle size: 98400434
vishakh@Vishakhs-MacBook-Pro malayalam-character-recognition %
```

Training

```
[vishakh@Vishakhs-MacBook-Pro malayalam-character-recognition % python3 train.py
  File "train.py", line 1
    ~~import keras
      ^
SyntaxError: invalid syntax
[vishakh@Vishakhs-MacBook-Pro malayalam-character-recognition % python3 train.py
  File "train.py", line 1
    ~~import keras
      ^
SyntaxError: invalid syntax
[vishakh@Vishakhs-MacBook-Pro malayalam-character-recognition % python3 train.py
Training set (16800, 32, 32) (16800,)
Validation set (2400, 32, 32) (2400,)
Training set (16800, 32, 32, 1) (16800, 48)
Validation set (2400, 32, 32, 1) (2400, 48)
Testing set (4800, 32, 32, 1) (4800, 48)
2022-04-17 11:13:24.438843: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions
in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

Image Input Scan

```
vishakh@Vishakhs-MacBook-Pro malayalam-character-recognition % python3 scan.py -i DRAWING-3.png
2022-04-17 11:26:22.578443: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions
in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
(1, 32, 32, 1)
[3333 3334 3335 3337 3342 3343 3346 3349 3350 3351 3352 3353 3354 3355
 3356 3357 3358 3359 3360 3361 3362 3363 3364 3365 3366 3367 3368 3370
 3371 3372 3373 3374 3375 3376 3377 3378 3379 3380 3381 3382 3383 3384
 3385 3450 3451 3452 3453 3454]
##########**##########
Imagefile =  DRAWING-3.png
Character =  3375
Confidence =  55.626821517944336 %
Other predictions
Character =  3375
Confidence =  55.626821517944336 %
Character =  3337
Confidence =  44.18770372867584 %
Character =  3362
Confidence =  0.17071510665118694 %
```

Image Input Scan

```
Confidence =  1.0384723081457992e-07 %
Character =  3383
Confidence =  9.934317257709324e-09 %
Character =  3350
Confidence =  5.369035077640305e-09 %
Character =  3343
Confidence =  3.482613503136278e-09 %
Character =  3374
Confidence =  2.0311273496442794e-09 %
Character =  3384
Confidence =  4.916649552766383e-10 %
Character =  3453
Confidence =  4.826779601008191e-10 %
Character =  3364
Confidence =  1.3173149637085713e-10 %
Character =  3351
Confidence =  7.516837087669093e-11 %
Character =  3450
Confidence =  6.822261939404273e-11 %
Character =  3377
Confidence =  6.034847101210605e-11 %
Character =  3382
Confidence =  1.0362264296409282e-11 %
Character =  3360
Confidence =  7.7307460427797e-13 %
Character =  3372
Confidence =  4.526555505071769e-13 %
Character =  3353
Confidence =  3.472338934193777e-13 %
Character =  3454
Confidence =  1.840139902636157e-13 %
Character =  3352
Confidence =  3.562473404177002e-14 %
Character =  3334
Confidence =  3.468233312558876 4e-14 %
Character =  3366
Confidence =  4.080718535379276e-15 %
Character =  3376
Confidence =  2.059332143963644e-15 %
Character =  3367
Confidence =  6.375256343236046e-16 %
Character =  3355
Confidence =  6.223678804708153e-16 %
Character =  3371
Confidence =  4.593098608622691e-16 %
Character =  3380
Confidence =  3.449243892672619e-16 %
Character =  3342
Confidence =  2.0325614360551005e-16 %
Character =  3381
Confidence =  2.6112336764436823e-17 %
Character =  3363
Confidence =  4.223664017001736e-18 %
Character =  3368
Confidence =  4.760434522662261e-19 %
Character =  3378
Confidence =  2.4005092376812137e-19 %
Character =  3365
Confidence =  5.813744322507426e-20 %
Character =  3385
Confidence =  9.694998959324688e-23 %
-0.00012880299998663247
vishakh@Vishakhs-MacBook-Pro malayalam-character-recognition %
```

Output


output.txt — Edited
പറവ

Future Works

- It will be used for converting whole malayalam documents.
- Practical applications like postal mail, lecture notes etc.
- Writer identification: It will be useful for various forensic and demographic investigations.
- Malayalam handwritten keyword spotting.
- The recognition of multi word, line, paragraph, page, etc.
- Medical applications (Ayurvedic Prescriptions)

Conclusion

- We have developed a recognition model that is able to recognize individual characters.
- The project is mainly comprised of two phases :
  - Segmentation
  - Recognition
- The segmentation phase segments the words from the input and the characters from the segmented words.
- The recognition phase recognizes the characters in their order of occurrence in the input image and stores it into an output text file.