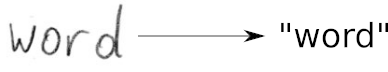
***Hand writing to digital text***

# Handwritten Text Recognition

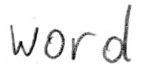
**Handwritten Text Recognition (HTR) system implemented with TensorFlow (TF) and** trained on the IAM off-line HTR dataset. The model takes **images of single words or text lines (multiple words) as input** and **outputs the recognized text**. 3/4 of the words from the validation-set are correctly recognized, and the character error rate is around 10%.

[](https://github.com/githubharald/SimpleHTR/blob/master/doc/htr.png)

## Run demo

* Download one of the pretrained models
  + [Model trained on word images](https://www.dropbox.com/s/mya8hw6jyzqm0a3/word-model.zip?dl=1): only handles single words per image, but gives better results on the IAM word dataset
  + [Model trained on text line images](https://www.dropbox.com/s/7xwkcilho10rthn/line-model.zip?dl=1): can handle multiple words in one image
* Put the contents of the downloaded zip-file into the model directory of the repository
* Go to the src directory
* Run inference code:
  + Execute python main.py to run the model on an image of a word
  + Execute python main.py --img\_file ../data/line.png to run the model on an image of a text line

The input images, and the expected outputs are shown below when the text line model is used.

[](https://github.com/githubharald/SimpleHTR/blob/master/data/word.png)

> python main.py

Init with stored values from ../model/snapshot-13

Recognized: "word"

Probability: 0.9806370139122009

[](https://github.com/githubharald/SimpleHTR/blob/master/data/line.png)

> python main.py --img\_file ../data/line.png

Init with stored values from ../model/snapshot-13

Recognized: "or work on line level"

Probability: 0.6674373149871826

import os

import cv2

import random

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

from keras import backend as K

from keras.models import Model

from keras.layers import Input, Conv2D, MaxPooling2D, Reshape, Bidirectional, LSTM, Dense, Lambda, Activation, BatchNormalization, Dropout

from keras.optimizers import Adam

## **Load and view data**

train = pd.read\_csv('/kaggle/input/handwriting-recognition/written\_name\_trai n\_v2.csv')

valid = pd.read\_csv('/kaggle/input/handwriting-recognition/written\_name\_validation\_v2.csv')

plt.figure(figsize=(15, 10))

for i **in** range(6):

ax = plt.subplot(2, 3, i+1)

img\_dir = '/kaggle/input/handwriting-recognition/train\_v2/train/'+train.loc[i, 'FILENAME']

image = cv2.imread(img\_dir, cv2.IMREAD\_GRAYSCALE)

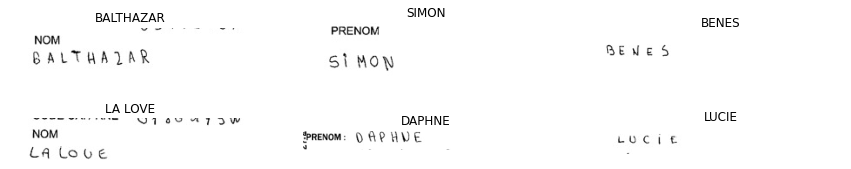
plt.imshow(image, cmap = 'gray')

plt.title(train.loc[i, 'IDENTITY'], fontsize=12)

plt.axis('off')

plt.subplots\_adjust(wspace=0.2, hspace=-0.8)

hand written to text:



Img recognization

If you’ve ever looked at an image file’s properties before, it’ll show the dimensions of the image, i.e. the height and width of the image. The height and width are based on number of pixels. For example, if the dimensions of an image are 400x300 (width x height), then the total number of pixels in the image is 120000.



Pixel data into image :

The function **tensorflow.io.read\_file** takes the file name as its required argument and returns the contents of the file as a tensor with type **tensorflow.string**. When the input file is an image, the output of **tensorflow.io.read\_file** will be the raw byte data of the image file. Although the raw byte output represents the image's pixel data, it cannot be used directly. Let’s first see the implementation in Python using the soccer ball image.

import tensorflow values = tf.io.read\_file('soccer\_ball.jpg')

we need to **resize the image** as for data augmentation. The function we use for resizing pixel data is **tensorflow.image.resize\_images**.

def **decode\_image**(filename, image\_type, resize\_shape, channels):  
value=tensorflow.io.read\_file(filename)  
if image\_type == 'png':

decoded\_image = tensorflow.image.decode\_png(value, channels=channels)

elif image\_type == 'jpeg':

decoded\_image = tensorflow.image.decode\_jpeg(value, channels=channels)

else:

decoded\_image = tensorflow.image.decode\_image(value, channels=channels

if resize\_shape is not None and image\_type in ['png', 'jpeg']:  
decoded\_image = tf.image.resize(decoded\_image, resize\_shape)

return decoded\_image

DATA SET:

The **Dataset** class makes it easier and more efficient to perform tasks with all the image files. After we create a dataset with the image files, we will need to decode each file’s contents into usable pixel data. Since the **decode\_image** function works for single image files, we will need to use the dataset object's **map** function to apply **decode\_image** to each image file in our dataset.

import tensorflow as tfdef **get\_dataset**(image\_paths, image\_type, resize\_shape, channels):  
 filename\_tensor = tf.constant(image\_paths)  
 dataset = tf.data.Dataset.from\_tensor\_slices(filename\_tensor)  
   
 def **\_map\_fn**(filename):  
 decode\_images = decode\_image(filename, image\_type, resize\_shape, channels=channels)  
 return decode\_images  
   
 map\_dataset = dataset.map(\_map\_fn) # we use the map method: allow to apply the function \_map\_fn to all the   
 # elements of dataset   
 return map\_datase

get\_image\_data(['soccer\_ball.jpg'],

'jpg', 0, 0)

O/p:



Summary:

Here we observe convertion of hand written documents to digital text .in image recognization we observe  it’ll show the dimensions of the image, i.e. the height and width of the image. The height and width are based on number of pixels.in first program thatwe can seen hand written data into text data i.e: *word-*word. Here we learn how to turn pixel data into image , it’ll show the dimensions of the image, i.e. the height and width of the image. The height and width are based on number of pixels