Documentation for OCR Pipeline Script

Introduction

This document provides detailed documentation for a Python script designed to load and process images from a dataset, perform Optical Character Recognition (OCR) on the images, and display the results. The script utilizes the libraries `numpy`, `cv2`, `matplotlib.pyplot`, and `os`.

# Prerequisites

Before running the script, ensure that you have the following libraries installed:

* numpy
* cv2 (OpenCV)
* matplotlib
* os (standard library)

You can install the necessary libraries using pip:

pip install numpy opencv-python matplotlib

# Script Overview

The script is divided into several key steps:

* Loading an image from the dataset
* Performing a simple OCR simulation
* Displaying the image and printing the recognized text

## Script Initialization

Import the required libraries:

python

import numpy as np

import cv2

import matplotlib.pyplot as plt

import os

# Function Definitions

## Step 1: Load an Image from the Dataset

The `load\_dataset\_image` function loads an image from the specified dataset directory. The function takes two parameters: `dataset\_path` (the path to the dataset directory) and `index` (the index of the image to load, default is 0).

python

def load\_dataset\_image(dataset\_path, index=0):

# List all image files in the dataset directory

image\_files = sorted([file for file in os.listdir(dataset\_path) if file.endswith(('png', 'jpg', 'jpeg'))])

if len(image\_files) == 0:

raise FileNotFoundError("No images found in the dataset directory!")

# Load the specified image

image\_path = os.path.join(dataset\_path, image\_files[index])

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

return img

This function lists all image files in the dataset directory and loads the specified image in grayscale mode. If no images are found, it raises a `FileNotFoundError`.

## Step 2: Perform OCR (Simulated)

The `simple\_ocr` function performs a simulated OCR by identifying black pixel regions in the image. It takes the loaded grayscale image as input and returns the extracted text.

python

def simple\_ocr(img):

# Extract text by identifying black pixel regions (dummy OCR simulation)

height, width = img.shape

extracted\_text = []

for i in range(0, height, 10): # Simulate line reading

line = "".join(["#" if pixel < 128 else " " for pixel in img[i, :]])

extracted\_text.append(line.strip())

return "\n".join(extracted\_text)

The function iterates through the image in steps of 10 pixels (simulating line reading) and checks each pixel's intensity to determine if it is part of the text (black). It constructs lines of text using "#" for black pixels and " " for white pixels.

## Step 3: Display and Process the Image

The `ocr\_pipeline` function orchestrates the OCR pipeline. It loads a sample image from the dataset, displays it using `matplotlib`, and performs OCR on the image.

python

def ocr\_pipeline(dataset\_path):

# Load a sample image from the dataset

img = load\_dataset\_image(dataset\_path, index=0) # You can change the index to load different images

# Show the image

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

plt.title("Sample Handwritten Image")

plt.axis("off")

plt.show()

# Perform OCR on the image

recognized\_text = simple\_ocr(img)

print("Recognized Text (Simulated OCR):")

print(recognized\_text)

This function uses `load\_dataset\_image` to load the first image from the dataset, displays the image using `matplotlib`, and prints the OCR results obtained from `simple\_ocr`.

# Main Execution

The script's main execution block sets the dataset path and calls the `ocr\_pipeline` function.

python

if \_\_name\_\_ == "\_\_main\_\_":

dataset\_path = "./kaggle\_dataset/handwritten\_images" # Update this path to your dataset directory

ocr\_pipeline(dataset\_path)

Update the `dataset\_path` variable to point to your dataset directory before running the script.

# Conclusion

This script provides a simple yet effective way to simulate OCR on handwritten images. By following the steps outlined in this documentation, you can easily adapt and expand the script for more complex OCR tasks or integrate it with other image processing workflows.

For any questions or further assistance, feel free to reach out to the script's author or refer to the official documentation of the libraries used.